5-2012

An Optimization Approach to a Geometric Packing Problem

Bradley Paynter
Clemson University, bradpaynter@gmail.com

An Optimization Approach to a Geometric Packing Problem

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mathematical Sciences

---

by
Bradley J. Paynter
May 2012

Accepted by:
Dr. Douglas R. Shier, Committee Chair
Dr. Warren P. Adams
Dr. Brian C. Dean
Dr. Daniel D. Warner

Abstract

We investigate several geometric packing problems (derived from an industrial setting) that involve fitting patterns of regularly spaced disks without overlap. We first derive conditions for achieving the feasible placement of a given set of patterns and construct a network formulation that, under certain conditions, allows the calculation of such a placement. We then discuss certain related optimization problems (e.g., fitting together the maximum number of patterns) and broaden the field of application by showing a connection to the well-known Periodic Scheduling Problem. In addition, a variety of heuristics are developed for solving large-scale instances of these provably difficult packing problems. The results of extensive computational testing, conducted on these heuristics, are presented.

Dedication

This work is dedicated to my wife, Angela, for the years of sacrifice that she has made to help me achieve this goal. I know that it would never have become a reality without her love and support.

Acknowledgments

The author wishes to thank Dr. R. Jamison for helping to lay some of the initial groundwork for this research. Also to Dr. K. James for his help with many of the Number Theory problems that arose during this project and Dr. B. Dean for his assistance with my understanding of NP-Completeness. Mostly I would like to thank Dr. D. Shier for the many ideas, suggestions and the uncountable hours spent turning my unguided ramblings into the perfect prose presented here.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

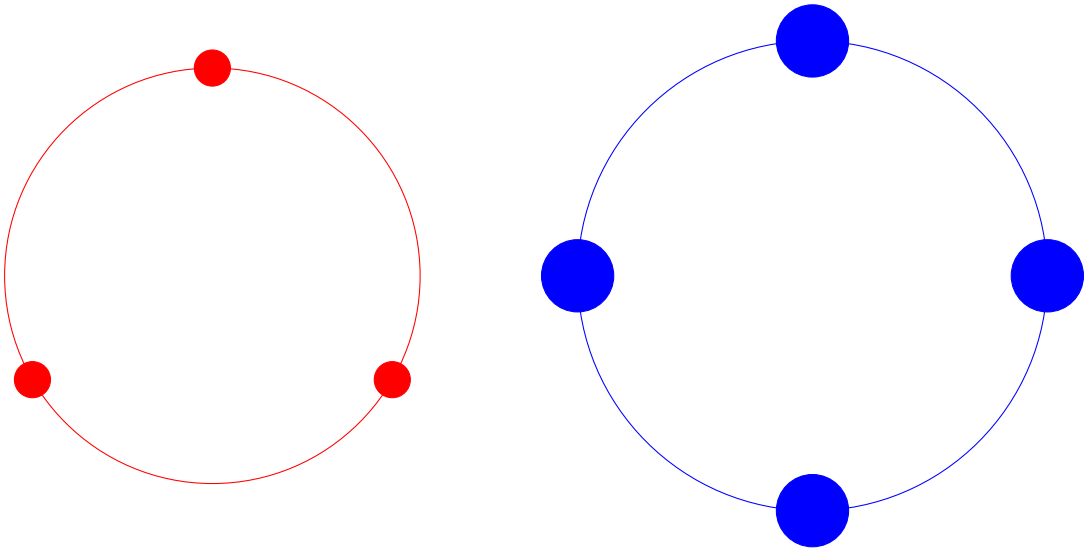
In this dissertation, we are concerned with the problem of fitting together objects that occur periodically. These objects can be intervals on the number line or shapes in the plane. This research was motivated by the following example from an industrial setting:

A company uses a machine to drill holes in wheels to attach the wheels to vehicles with lug nuts. The holes are arranged in patterns. Each pattern has a specific number of holes, all with the same inner diameter, whose centers lie, evenly spaced, around the outer circumference of a larger circle. The factory uses templates to guide the machine in drilling patterns of holes in multiple wheel types. A template can contain multiple patterns whose outer circles are concentric. The company wishes to construct a solution that uses the fewest number of templates to accommodate a given set of patterns.

To illustrate this motivating problem, we present the following simplified example.

Example 1.0.1: We are given a set $\mathcal{P}^1$ containing two patterns, the first (P_1) comprising three holes, each of radius 1 unit, evenly spaced around a circle of radius 11.5 units, and the second (P_2) comprising four holes, each of radius 2 units, evenly spaced around a circle of radius 13 units. Can we fit both patterns without overlap on the same template? These patterns are shown in Figure 1.0.1(a) and Figure 1.0.1(b).

We can arbitrarily designate a hole of each pattern as the first and then number the remaining



1.0.1(a): Pattern P_1

1.0.1(b): Pattern P_2

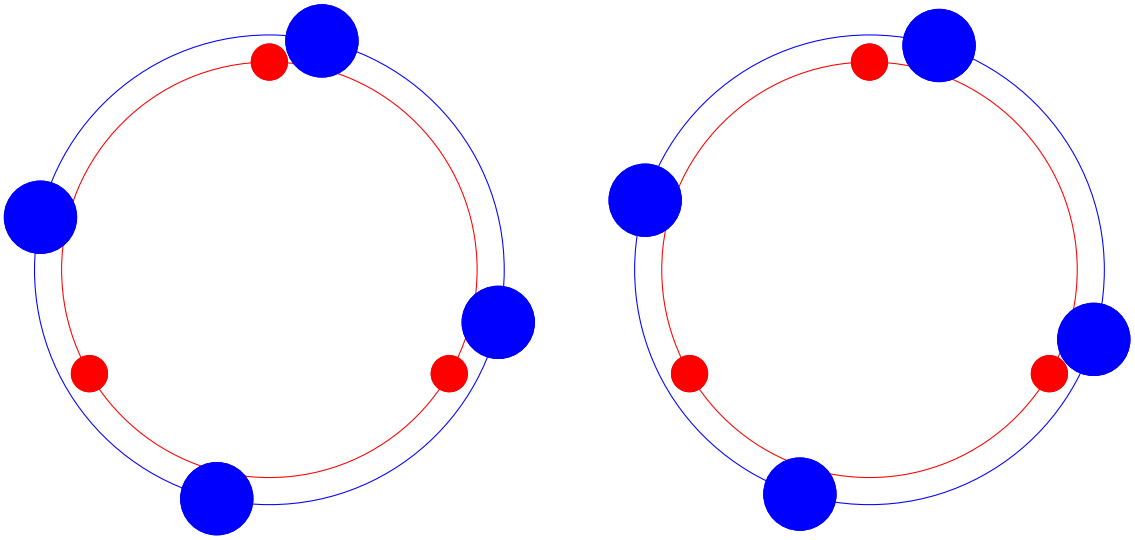
Figure 1.0.1: $\mathcal{P}^1$: Patterns

holes sequentially in a clockwise direction. We can now arrange these patterns so that they do not overlap with the first hole of P_2 lying immediately after (in a clockwise sense) the first hole of P_1, as shown in Figure 1.0.2(a). Notice that we can continue to rotate P_2 clockwise and the patterns will not overlap until the second hole of P_2 touches the second hole of P_1, as shown in Figure 1.0.2(b).

In this example, the patterns can fit together. Moreover, there is a range of relative displacements between these two patterns such that they can be feasibly arranged.

Another example of fitting together periodic objects is found in periodic scheduling. Burkhart [7] describes this in a planar setting where two regular polygons are inscribed inside a circle. The problem is then to maximize the minimum distance between any two corners of these polygons. An application of this occurs when regularly operating trains share a track. If the trains arrive with different periodicities, we wish to maximize the minimum time between trains for safety reasons. These ideas have been extended to multiple polygons by Vince [27].

By far the most common instances of periodic scheduling involve events on the number line which reoccur at regular intervals over an infinite horizon. These problems originally arose in the schedul-



1.0.2(a): Feasible solution 1

1.0.2(b): Feasible solution 2

Figure 1.0.2: $\mathcal{P}^1$: Solutions

ing of tasks on a computer where several types of requests are submitted [5, 10, 20]. A request is an input that requires some processing on the computer. An example of this occurs where a sensor provides a reading once every five seconds and every reading needs to be analyzed by the processor. In general, each type of request occurs at known regular intervals. Each request takes a certain amount of time to process and has to be dealt with within some fixed time period (usually before the next occurrence of the same type of request). The problem is then to specify a schedule for the processor that identifies which request to process and when. These problems usually allow preemption of tasks. That is, a task can be halted part way through processing so that another job can be processed; the original task can be resumed later without penalty.

The next development was to enforce that each request has to be processed immediately. The requests then become events that occur with a fixed period. This is commonly called periodic assignment: the problem is to assign the incoming requests to the minimum possible number of processors so that all requests are handled. Orlin [22] showed that, when all events have the same period and any processor can handle any event, periodic assignment can be solved in polynomial time. This specific problem arises in the arena of airline scheduling where regularly scheduled flights must be covered by equipment and crews. This problem can be extended to situations where each

occurrence of the same event must be handled by the same processor (known as constrained periodic assignment); in this case periodic assignment becomes NP-hard since it is equivalent to coloring circular-arc graphs [19, 22]. Alternatively, the requirement that all events have the same period can be relaxed [19, 23]; in this case the problem is known as unconstrained periodic assignment which is also NP-hard.

In some situations, events will occur periodically once they are started. Choosing the start time of a periodic event thus determines the times of future occurrences of the same event [18, 28]. The problem is then to find offsets such that a set of periodic events do not overlap. This problem is known as constrained periodic scheduling since we wish to determine a schedule that governs when the events occur. This formulation approximates the problem of fitting regularly-spaced patterns of holes discussed in Example 1.0.1 above and is of major interest in this dissertation (Chapter 3). It also has applications in the design of airplane computer systems [11]. Heuristics for this formulation have been developed [28] for the special case where all events have integer periods and take one time unit to process. Additionally, heuristics have been developed [18] for the case where the periods of the events are harmonic (i.e., events can be ordered so that the period of event i divides the period of event j for $i < j$).

Significant research has also addressed the problem of scheduling periodic tasks where the period is a decision variable as well [2, 4, 13]. The aim of these problems is typically to minimize some function of the period lengths. This formulation has a number of applications including the dissemination of data via unidirectional broadcast [3]. In this application, customers arrive according to some random distribution and wish to access certain data. The entity responsible for transmitting that data does not have the bandwidth to process all requests on demand, so it transmits all data that could possibly be requested on a periodic schedule. The goal is to find a transmission schedule that minimizes the wait time for the customers. Another application involves the replenishment of stock, where a store may need to schedule periodic deliveries by their suppliers so as to never run out of stock. Yet another example is that of performing routine maintenance on equipment, where the cost of running that equipment is proportional to the length of time that has passed since the last maintenance was performed.

Serafini and Ukovich [26] propose a more general model for solving periodic scheduling problems that allows for several of the different modifications listed above. Their approach involves a network formulation which includes a node for each occurrence of each event, and they provide a pseudo-polynomial time solution algorithm.

This dissertation is organized as follows. In Chapter 2 we formulate the problem of fitting patterns of disks without overlap and prove various feasibility criteria for this problem. We also introduce mixed integer programming and network algorithms as techniques for finding offsets that lead to non-overlapping patterns. Chapter 3 shows how the general problem of fitting patterns together without overlap can be approximated by the problem of scheduling periodic maintenance. Next, the complexities of the various problems discussed are established in Chapter 4. Finally, we develop several heuristics and detail their performance in Chapters 5 and 6.

Chapter 2

Problem Formulation and Development

In this chapter we establish mathematical requirements that govern whether patterns overlap or not. We start by formally defining patterns and detailing some of a pattern's basic properties. We then derive feasibility conditions for two patterns to fit without overlap, and extend those results to multiple patterns. Next, we define a set of problems relating to the distribution of such patterns onto feasible templates and formulate those problems as mixed integer programming problems. Finally, we relate the derived feasibility conditions to the dual feasibility conditions for the well-studied shortest path problem in networks. This results in our defining an appropriate network and considering the conditions under which this network can be used to guide algorithms for the non-overlapping placement of patterns.

2.1 Definitions and Properties

We define a pattern to be a set of open disks on the plane. Each pattern P consists of n disks of radius ρ, arranged so that their centers are evenly spaced around the circumference of a circle of radius R. To eliminate the need to account for the uninteresting special case when $n = 1$ we will assume throughout this dissertation that $n > 1$. The disks themselves will be indexed by the set of residues $\mathbb{Z}/n\mathbb{Z}$ but can be referred to without confusion by any element of the equivalence class.

The first disk of the pattern will be that indexed by zero and the angle of rotation of the center of that first disk will be denoted as φ and measured in radians.

Definition 2.1.1 *A pattern P maps a starting angle of rotation φ to a set of disks on the plane: i.e.,*

$$P(\varphi) = \bigcup_{k\in\mathbb{Z}} P(\varphi, k) \subseteq \mathbb{R}^2.$$

Each disk $P(\varphi, k)$ is centered at $\left(R, \varphi + k\frac{2\pi}{n}\right)$, has radius ρ and forms the open set

$$P(\varphi, k) = \left\{(r, \theta)\middle| r^2 - 2rR\cos\left(\theta - \varphi - k\frac{2\pi}{n}\right) + R^2 < \rho^2\right\}.$$

The mapping is completely defined by the strictly positive constants $R, \rho \in \mathbb{R}$ and $n \in \mathbb{Z}$, and thus can also be designated as $P = (R, n, \rho)$.

Throughout this dissertation we will reference certain variables either as unknowns or as given fixed values. To differentiate between these, we will typically denote unknowns as unadorned symbols (i.e., φ) and fixed, known values will be given bars (i.e., $\overline{\varphi}$) or some similar marking.

Definition 2.1.2 *A template is a set of patterns $\mathcal{P} = \{P_i \mid i \in I\}$ with fixed starting angles $\Phi = \{\overline{\varphi}_i \mid i \in I\}$ and will be denoted $\mathcal{T} = \mathcal{P}(\Phi) = \{P_i(\overline{\varphi}_i) \mid i \in I\}$. Furthermore, if*

$$P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) = \emptyset, \quad \forall i \neq j$$

then the template $\mathcal{T}$ is referred to as a feasible template and the starting positions Φ are said to be valid for $\mathcal{P}$.

The collection of all starting positions for which $\mathcal{P}$ is a feasible template is defined as

$$\mathcal{S}(\mathcal{P}) = \{\Phi \mid \mathcal{P}(\Phi) \text{ is a feasible template}\}.$$

The following lemma details some of the basic symmetrical properties of a pattern which we will exploit later.

Lemma 2.1.3 *A pattern $P = (R, n, \rho)$ has the following immediate properties:*

(a) Disks $P(\varphi, k)$ and $P(\varphi, k+1)$ will overlap unless $\rho \leq R \sin\left(\frac{\pi}{n}\right)$. Since this is undesirable, we will assume that this holds for all given patterns.

(b) $P(\varphi) = P\left(\varphi + \ell\frac{2\pi}{n}\right)$ for all $\ell \in \mathbb{Z}$ and for all $\varphi \in \mathbb{R}$.

(c) If point $(r, \theta) \in P(\varphi)$ then

$$\left(r, \theta + \ell\frac{2\pi}{n}\right) \in P(\varphi), \quad \forall \ell \in \mathbb{Z}.$$

(d) If point $(r, \theta) \in P(\varphi)$ then $R - \rho < r < R + \rho$.

Proof: Let $P = (R, n, \rho)$ be a given pattern.

(a) Figure 2.1.1 shows two consecutive disks of pattern $P = (R, n, \rho)$ (drawn in blue and centered at C_1 and C_2) and the circle the centers of P lie on (drawn in black and centered at O). In order for the blue disks to be disjoint, we need the line segment $\overline{C_1C_2}$ (drawn in red) to be longer than twice the radius of the disks. Recall that the line segments $\overline{OC_1}$ and $\overline{OC_2}$ have length R and the minor angle $\angle C_1OC_2$ has measure $\frac{2\pi}{n}$. Then the disks will be disjoint iff

$$\begin{aligned}
(2\rho)^2 &\leq R^2 + R^2 - 2RR\cos\left(\frac{2\pi}{n}\right) \\
4\rho^2 &\leq 2R^2 - 2R^2\cos\left(\frac{2\pi}{n}\right) \\
4\rho^2 &\leq 2R^2\left(1 - \cos\left(\frac{2\pi}{n}\right)\right) \\
4\rho^2 &\leq 4R^2\left(\frac{1 - \cos\left(2\frac{\pi}{n}\right)}{2}\right) \\
\rho^2 &\leq R^2\sin^2\left(\frac{\pi}{n}\right) \\
\rho &\leq R\sin\left(\frac{\pi}{n}\right)
\end{aligned}$$

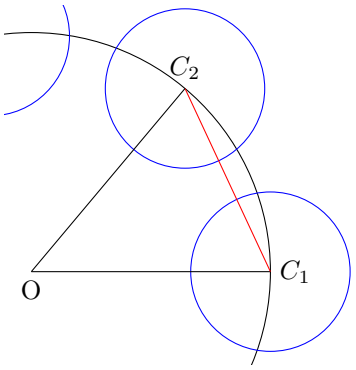


Figure 2.1.1: Non-overlapping requirement

(b) By definition

$$\begin{aligned} P\left(\varphi+\ell\frac{2\pi}{n}\right) &= \bigcup_{k\in\mathbb{Z}}\left\{(r,\theta)\middle| r^2-2rR\cos\left(\theta-\left(\varphi+\ell\frac{2\pi}{n}\right)-k\frac{2\pi}{n}\right)+R^2<\rho^2\right\} \\ &= \bigcup_{k\in\mathbb{Z}}\left\{(r,\theta)\middle| r^2-2rR\cos\left(\theta-\varphi-(\ell+k)\frac{2\pi}{n}\right)+R^2<\rho^2\right\} \\ &= \bigcup_{k'\in\mathbb{Z}}\left\{(r,\theta)\middle| r^2-2rR\cos\left(\theta-\varphi-k'\frac{2\pi}{n}\right)+R^2<\rho^2\right\} \\ &= P(\varphi) \end{aligned}$$

where $k'=k+\ell$.

(c) Let $(r,\theta)\in P(\varphi)$ be given. Then there exists $k\in\mathbb{Z}$ such that $(r,\theta)\in P(\varphi,k)$. Thus

$$\begin{aligned} r^2-2rR\cos\left(\theta-\varphi-k\frac{2\pi}{n}\right)+R^2<\rho^2 \\ r^2-2rR\cos\left(\theta+\ell\frac{2\pi}{n}-\varphi-k\frac{2\pi}{n}-\ell\frac{2\pi}{n}\right)+R^2<\rho^2 \\ r^2-2rR\cos\left(\left(\theta+\ell\frac{2\pi}{n}\right)-\varphi-(k+\ell)\frac{2\pi}{n}\right)+R^2<\rho^2 \end{aligned}$$

Thus $\left(r,\theta+\ell\frac{2\pi}{n}\right)\in P(\varphi,k+\ell)\subset P(\varphi)$.

(d) Let $(r, \theta) \in P(\varphi)$. Then there exists $k \in \mathbb{Z}$ such that

$$\begin{aligned} r^2 - 2rR\cos\left(\theta - \varphi - k\frac{2\pi}{n}\right) + R^2 &< \rho^2 \\ r^2 - 2rR + R^2 &< \rho^2 \\ (r-R)^2 &< \rho^2 \\ |r-R| &< \rho \\ -\rho < r - R &< \rho \\ R - \rho < r &< R + \rho \end{aligned}$$

■

As discussed in Chapter 1, a fundamental problem of interest in this dissertation is that of partitioning a set of patterns $\mathcal{P} = \{P_i \mid i \in I\}$ into the fewest number of feasible templates. To achieve this, we will partition the index set I into a collection of disjoint sets $\mathcal{I} = \{I_u \mid u \in U\}$; accordingly the set of patterns $\mathcal{P}$ is partitioned into sets $\mathcal{P}_u = \{P_i \mid i \in I_u\}$ so that the templates $\mathcal{T}_u = \{P_i(\overline{\varphi}_i) \mid i \in I_u\}$ are feasible. This problem can be stated as the Minimum Templates Problem (MinTemp):

MinTemp($\mathcal{P}$):

Given a set of patterns $\mathcal{P} = \{P_i \mid i \in I\}$, determine a partition of I, $\mathcal{I} = \{I_u \mid u \in U\}$, with minimum $|U|$, such that valid starting positions exist for each $\mathcal{P}_u = \{P_i \mid i \in I_u\}$.

It is easy to see a connection between this problem and the well-known Bin Packing Problem [6]. However, whereas it is relatively simple to determine whether or not an item fits in a bin, it is more complicated to determine whether a set of patterns can form a feasible template. As a result, we focus first on this subproblem, the Feasible Fit Problem (FFP):

FFP($\mathcal{P}$):

Given a set of patterns $\mathcal{P} = \{P_i \mid i \in I\}$, does there exist a set of starting positions Φ such that $\mathcal{P}(\Phi)$ is a feasible template?

If it is not possible to fit all patterns in a set on a single template, we want to determine a largest possible feasible template that can be formed. This problem is denoted as the Maximum Template Problem (MaxTemp):

MaxTemp($\mathcal{P}$)**:**
Given a set of patterns $\mathcal{P} = \{P_i \mid i \in I\}$, determine a largest subset $I_1 \subseteq I$ such that valid starting positions exist for $\mathcal{P}_1 = \{P_i \mid i \in I_1\}$.

With the goal of solving these three problems, we devote the rest of this chapter to deriving template feasibility conditions and using them to formulate the problems in ways that lead to solution techniques.

2.2 Two Patterns

We first investigate conditions under which two patterns $P_i(\varphi_i)$ and $P_j(\varphi_j)$ fit on the same feasible template. We could compare all possible pairs of disks $P_i(\varphi_i, k_i)$ and $P_j(\varphi_j, k_j)$, but this would be cumbersome. Alternatively, we could look at the disks of $P_j(\varphi_j)$ independently. For $P_j(\varphi_j, k_j)$ not to overlap with any disk of $P_i(\varphi_i)$, it must lie strictly "between" two consecutive disks of $P_i(\varphi_i)$. Unfortunately this approach is still algorithmically unwieldy, but it will form the theoretical basis for our approach. To start we consider a simple case.

Lemma 2.2.1 *Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given such that $n_i \mid n_j$. Let $\overline{\varphi}_i, \overline{\varphi}_j \in \mathbb{R}$ be given. Then $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is feasible iff $P_i(\overline{\varphi}_i, k) \cap P_j(\overline{\varphi}_j) = \emptyset$ for any $k \in \mathbb{Z}$.*

Proof: ($\Rightarrow$)
If $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is feasible then $P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) = \emptyset$ and this implies that $P_i(\overline{\varphi}_i, k) \cap P_j(\overline{\varphi}_j) = \emptyset$ for all $k \in \mathbb{Z}$.

($\Leftarrow$)
Assume that $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is not feasible and thus $P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) \neq \emptyset$. Let $k \in \mathbb{Z}$ be given and let $(\overline{r}, \overline{\theta}) \in P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j)$. Then there exists $k_i, k_j \in \mathbb{Z}$ such that $(\overline{r}, \overline{\theta}) \in P_i(\overline{\varphi}_i, k_i)$ and $(\overline{r}, \overline{\theta}) \in P_j(\overline{\varphi}_j, k_j)$.
Consider the set

$$I := \left\{ \left(\overline{r}, \overline{\theta} + k\frac{2\pi}{n_i} \right) \middle| k \in \mathbb{Z} \right\}.$$

From Property 2.1.3(c) and the fact that $(\overline{r}, \overline{\theta}) \in P_i(\overline{\varphi}_i)$, we get $I \subseteq P_i(\overline{\varphi}_i)$. Additionally, we have

$$I = \left\{ \left(\overline{r}, \overline{\theta} + k\frac{n_j}{n_i} \cdot \frac{2\pi}{n_j} \right) \middle| k \in \mathbb{Z} \right\}$$

and since $n_i \mid n_j$, $k\dfrac{n_j}{n_i} \in \mathbb{Z}$ for all k, so we again get $I \subseteq P_j(\overline{\varphi}_j)$ from Property 2.1.3(c). Thus $I \subseteq P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j)$. In fact $\left(\overline{r}, \overline{\theta} + k\dfrac{2\pi}{n_i} \right) \in P_i(\overline{\varphi}_i, k_i + k)$ for all $k \in \mathbb{Z}$. Specifically for $k = k - k_i$, this implies that $\left(\overline{r}, \overline{\theta} + (k - k_i)\dfrac{2\pi}{n_i} \right)$, which is an element of $I \subseteq P_j(\overline{\varphi}_j)$, is contained in $P_i(\overline{\varphi}_i, k)$. Thus $P_j(\overline{\varphi}_j) \cap P_i(\overline{\varphi}_i, k) \neq \emptyset$. ■

Before proceeding, we introduce the following notation common in number theory [16]:

$$(a, b) = \gcd(a, b) \qquad\qquad [a, b] = \operatorname{lcm}(a, b)$$

Lemma 2.2.2 *Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given. Define $P_{ij} = \big(R_i, [n_i, n_j], \rho_i\big)$. Then $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is a feasible template iff $\{P_{ij}(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is a feasible template.*

Proof: ($\Leftarrow$)

Assume that $P_{ij}(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) = \emptyset$. Consider

$$\begin{aligned}
P_i(\overline{\varphi}_i) &= \left\{ (r, \theta) \middle| r^2 - 2rR_i \cos\left(\theta - \overline{\varphi}_i - k\frac{2\pi}{n_i} \right) + R_i^2 < \rho_i^2,\ k \in \mathbb{Z} \right\} \\
&= \left\{ (r, \theta) \middle| r^2 - 2rR_i \cos\left(\theta - \overline{\varphi}_i - k\frac{2\pi n_j}{(n_i, n_j)[n_i, n_j]} \right) + R_i^2 < \rho_i^2,\ k \in \mathbb{Z} \right\} \\
&= \left\{ (r, \theta) \middle| r^2 - 2rR_i \cos\left(\theta - \overline{\varphi}_i - k\frac{n_j}{(n_i, n_j)}\frac{2\pi}{[n_i, n_j]} \right) + R_i^2 < \rho_i^2,\ k \in \mathbb{Z} \right\}
\end{aligned}$$

Since $(n_i, n_j) \mid n_j$ we have $k\dfrac{n_j}{(n_i, n_j)} \in \mathbb{Z}$ for all $k \in \mathbb{Z}$. Thus

$$P_i(\overline{\varphi}_i) \subseteq P_{ij}(\overline{\varphi}_i).$$

So if $P_{ij}(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) = \emptyset$ then $P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) = \emptyset$ and thus $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is a feasible template.

($\Rightarrow$)

Let $\overline{\varphi}_i$ and $\overline{\varphi}_j$ be given such that $P_j(\overline{\varphi}_j) \cap P_{ij}(\overline{\varphi}_i) \neq \emptyset$. We wish to show that $P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) \neq \emptyset$. Let $(\overline{r}, \overline{\theta})$ be any point in $P_j(\overline{\varphi}_j) \cap P_{ij}(\overline{\varphi}_i)$ and define

$$J = \left\{ \left(\overline{r}, \overline{\theta} + k_j \frac{2\pi}{n_j} \right) \middle| k_j \in \mathbb{Z} \right\}.$$

Since $(\overline{r}, \overline{\theta}) \in P_j(\overline{\varphi}_j)$, Property 2.1.3(c) gives $J \subseteq P_j(\overline{\varphi}_j)$. Similarly, if we define

$$H = \left\{ \left(\overline{r}, \overline{\theta} + k \frac{2\pi}{[n_i, n_j]} \right) \middle| k \in \mathbb{Z} \right\}$$

then $H \subseteq P_{ij}(\overline{\varphi}_i)$. Furthermore, consider $(\overline{r}, \hat{\theta}) \in H \cap P_i(\overline{\varphi}_i)$.[1] Since $(\overline{r}, \hat{\theta}) \in H$ there exists $\hat{k} \in \mathbb{Z}$ such that

$$\hat{\theta} = \overline{\theta} + \hat{k} \frac{2\pi}{[n_i, n_j]}.$$

Define

$$I = \left\{ \left(\overline{r}, \overline{\theta} + \hat{k} \frac{2\pi}{[n_i, n_j]} + k_i \frac{2\pi}{n_i} \right) \middle| k_i \in \mathbb{Z} \right\}.$$

Since $\left(\overline{r}, \overline{\theta} + \hat{k} \frac{2\pi}{[n_i, n_j]} \right) \in P_i(\overline{\varphi}_i)$, Property 2.1.3(c) gives $I \subseteq P_i(\overline{\varphi}_i)$.
Let $\overline{k}_i, \overline{k}_j \in \mathbb{Z}$ solve the diophantine equation

$$\hat{k}(n_i, n_j) = \overline{k}_j n_i - \overline{k}_i n_j. \tag{2.2.1}$$

We know that such $\overline{k}_i, \overline{k}_j$ exist since $(n_i, n_j) \mid \hat{k}(n_i, n_j)$. Thus

$$\begin{aligned} \hat{k}(n_i, n_j) &= \overline{k}_j n_i - \overline{k}_i n_j \\ \hat{k} \frac{n_i n_j}{[n_i, n_j]} + \overline{k}_i n_j &= \overline{k}_j n_i \\ \hat{k} \frac{2\pi}{[n_i, n_j]} + \overline{k}_i \frac{2\pi}{n_i} &= \overline{k}_j \frac{2\pi}{n_j}. \end{aligned}$$

This gives

$$\left(\overline{r}, \overline{\theta} + \hat{k} \frac{2\pi}{[n_i, n_j]} + \overline{k}_i \frac{2\pi}{n_i} \right) = \left(\overline{r}, \overline{\theta} + \overline{k}_j \frac{2\pi}{n_j} \right).$$

[1] Recall that P_{ij} was defined so that some of the disks of P_{ij} are also disks of P_i. Since H has one point in each disk in $P_{ij}(\overline{\varphi}_i)$ at least one of those points is in a disk of $P_i(\overline{\varphi}_i)$ and so $H \cap P_i(\overline{\varphi}_i) \neq \emptyset$.

From (2.2.1)

$$\left(\overline{r}, \overline{\theta} + \overline{k}_j \frac{2\pi}{n_j}\right) = \left(\overline{r}, \overline{\theta} + \hat{k}\frac{2\pi}{[n_i, n_j]} + \overline{k}_i \frac{2\pi}{n_i}\right) \in I$$

and we also know

$$\left(\overline{r}, \overline{\theta} + \overline{k}_j \frac{2\pi}{n_j}\right) \in J.$$

This implies that $I \cap J \neq \emptyset$. Finally, since $I \subseteq P_i(\overline{\varphi}_i)$ and $J \subseteq P_j(\overline{\varphi}_j)$, $P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) \neq \emptyset$ as desired. ■

Combining the previous two lemmas, we can now check that two patterns $P_i(\overline{\varphi}_i)$ and $P_j(\overline{\varphi}_j)$ do not overlap simply by forming the pattern $P_{ij} = (R_i, [n_i, n_j], \rho_i)$ and ensuring that $P_j(\overline{\varphi}_j, 0) \cap P_{ij}(\overline{\varphi}_i) = \emptyset$. This will form the basis of our approach for finding valid starting positions, but first we need to deal with another special case.

Lemma 2.2.3 *Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be such that $|R_i - R_j| \geq \rho_i + \rho_j$. Then any $\varphi_i, \varphi_j \in \mathbb{R}$ are valid for the template $\{P_i, P_j\}$.*

Proof: Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ satisfy $|R_i - R_j| \geq \rho_i + \rho_j$. Assume that $P_i(\varphi_i, k) \cap P_j(\varphi_j, \ell) \neq \emptyset$ and let $(\overline{r}, \overline{\theta}) \in P_i(\varphi_i, k) \cap P_j(\varphi_j, \ell)$ be given. Then

$$\begin{aligned}
\overline{r}^2 - 2\overline{r}R_i \cos\left(\overline{\theta} - \varphi_i - k\frac{2\pi}{n_i}\right) + R_i^2 &< \rho_i^2 & \qquad \overline{r}^2 - 2\overline{r}R_j \cos\left(\overline{\theta} - \varphi_j - \ell\frac{2\pi}{n_j}\right) + R_j^2 &< \rho_j^2 \\
\overline{r}^2 - 2\overline{r}R_i + R_i^2 &< \rho_i^2 & \overline{r}^2 - 2\overline{r}R_j + R_j^2 &< \rho_j^2 \\
(\overline{r} - R_i)^2 &< \rho_i^2 & (\overline{r} - R_j)^2 &< \rho_j^2 \\
|\overline{r} - R_i| &< |\rho_i| & |\overline{r} - R_j| &< |\rho_j| \\
|R_i - \overline{r}| &< \rho_i & |\overline{r} - R_j| &< \rho_j
\end{aligned}$$

and so, from the triangle inequality, we get

$$\begin{aligned}
|R_i - R_j| &= |R_i - \overline{r} + \overline{r} - R_j| \\
&\leq |R_i - \overline{r}| + |\overline{r} - R_j| \\
&< \rho_i + \rho_j.
\end{aligned}$$

Thus, if there exist $\varphi_i, \varphi_j \in \mathbb{R}$ such that $\{P_i(\varphi_i), P_j(\varphi_j)\}$ is not feasible, then $|R_i - R_j| < \rho_i + \rho_j$. Equivalently, if $|R_i - R_j| \geq \rho_i + \rho_j$ then $\{P_i(\varphi_i), P_j(\varphi_j)\}$ is feasible for all $\varphi_i, \varphi_j \in \mathbb{R}$. ■

Before continuing, it is useful to introduce the following notation:

$$\beta_{ij} = \frac{2\pi}{[n_i, n_j]}$$

and

$$\delta_{ij} = \begin{cases} 2 \arcsin\left(\sqrt{\dfrac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}}\right), & |R_i - R_j| < \rho_i + \rho_j \\ 0, & |R_i - R_j| \geq \rho_i + \rho_j. \end{cases}$$

To show that δ_{ij} is well defined, first note that since $|R_i - R_j| < \rho_i + \rho_j$, the operand of the square root is non-negative. Also note that Property 2.1.3(a) implies that

$$\rho_i \leq R_i \sin\left(\frac{\pi}{n_i}\right) \leq R_i$$

and similarly $\rho_j \leq R_j$. Thus

$$\begin{aligned} \sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} &\leq \sqrt{\frac{(R_i + R_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\ &= \sqrt{\frac{4R_iR_j}{4R_iR_j}} \\ &= 1. \end{aligned}$$

Thus the operand of the arcsine in the definition of δ_{ij} is in the traditionally defined domain $[-1, 1]$. This results in $0 \leq \delta_{ij} \leq 2\pi$ for all i, j.

Finally, we can combine the previous three lemmas into the following main result that characterizes the valid starting positions for any pair of patterns. The proof of this theorem is fairly lengthy and can be found in Appendix A.

Theorem 2.2.4 *Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given. Then $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is feasible iff there exists $k \in \mathbb{Z}$ such that*

$$k\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - \delta_{ij}. \tag{2.2.2}$$

Moreover, if $\delta_{ij} > 0$ then k is unique.

Comment 2.2.5: The proof of Theorem 2.2.4 shows that k from inequality (2.2.2) has meaning in the physical problem as well. Recall that, when a pattern was defined as a set of open disks, those disks were indexed by $k \in \mathbb{Z}$ and the disk arbitrarily designated as first corresponds to the disk where $k = 0$. Suppose we have two patterns, $P_i(\overline{\varphi}_i)$ and $P_j(\overline{\varphi}_j)$. We then form the pattern $P_{ij} = (R_i, [n_i, n_j], \rho_i)$ whose disks are indexed by k_{ij} and $P_{ij}(\overline{\varphi}_i, 0) = P_i(\overline{\varphi}_i, 0)$. Then $P_j(\overline{\varphi}_j, 0)$ must lie between two disks of pattern P_{ij}. These disks are $P_{ij}(\overline{\varphi}_i, k)$ and $P_{ij}(\overline{\varphi}_i, k+1)$. Thus k is just the relative offset between the indices of P_j and P_{ij}. To illustrate, we offer the following example.

Example 2.2.6: Recall the set $\mathcal{P}^1$ from Example 1.0.1, with given patterns $P_1 = (11.5, 3, 1)$ and $P_2 = (13, 4, 2)$. To start, we calculate the following values:

$$\beta_{12} = \frac{2\pi}{[n_1, n_2]} = \frac{2\pi}{[3,4]} = \frac{\pi}{6} \approx 0.524$$

$$\delta_{12} = 2\arcsin\sqrt{\frac{(\rho_1+\rho_2)^2 - (R_1 - R_2)^2}{4R_1R_2}} = 2\arcsin\sqrt{\frac{(1+2)^2 - (11.5-13)^2}{4(11.5)(13)}} = 2\arcsin\sqrt{\frac{6.75}{598}} \approx 0.213$$

We use inequality (2.2.2) to verify that $\overline{\varphi}_1 = 0$ and $\overline{\varphi}_2 = 0.75$ are valid starting positions. We will show later in Corollary 2.2.9 how to calculate k, but for now we assert the value $k = 1$. Then

$$\begin{aligned} k\beta_{12} + \delta_{12} &= 1(0.524) + 0.213 & (k+1)\beta_{12} - \delta_{12} &= (1+1)0.524 - 0.213 \\ &= 0.737 & &= 0.835 \end{aligned}$$

giving

$$0.737 \leq 0.75 \leq 0.835 \tag{2.2.3}$$
$$k\beta_{12} + \delta_{12} \leq \overline{\varphi}_2 - \overline{\varphi}_1 \leq (k+1)\beta_{12} - \delta_{12}.$$

Notice that there are a range of starting positions that would lead to a feasible template. In fact (2.2.3) shows that any starting positions that satisfy $0.737 \leq \varphi_2 - \varphi_1 \leq 0.835$ would be valid for P_1, P_2.

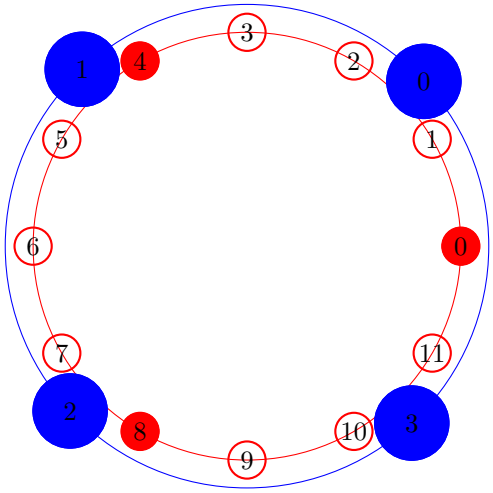


Figure 2.2.1: $\mathcal{P}^1$: The feasible template $\{P_1(0), P_2(0.75)\}$

Consider Figure 2.2.1 which contains the following:

- $P_1(0)$ shown as solid red disks,
- $P_2(0.75)$ shown as solid blue disks,
- $P_{12}(0)$ shown as both solid red disks and open red disks.

Both $P_2(0.75)$ and $P_{12}(0)$ have been labelled with their indexing integers. Notice that $P_2(0.75)$ does not overlap with either $P_1(0)$ or $P_{12}(0)$ as the calculations predicted. Also, the geometric interpretation of $k = 1$ is that $P_2(0.75, 0)$ lies between $P_{12}(0, 1)$ and $P_{12}(0, 2)$.

To remove any discomfort about the asymmetry of Theorem 2.2.4, we have the following corollary.

Corollary 2.2.7 *Suppose $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$. If there exists $k \in \mathbb{Z}$ such that*

$$k\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - \delta_{ij}$$

for some $\overline{\varphi}_i, \overline{\varphi}_j \in \mathbb{R}$, then

$$k'\beta_{ji} + \delta_{ji} \leq \overline{\varphi}_i - \overline{\varphi}_j \leq (k'+1)\beta_{ji} - \delta_{ji}$$

where $k' = -k - 1$.

Proof:

$$\begin{aligned}
k\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i &\leq (k+1)\beta_{ij} - \delta_{ij} \\
(-k'-1)\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i &\leq ((-k'-1)+1)\beta_{ij} - \delta_{ij} \\
(k'+1)\beta_{ij} - \delta_{ij} \geq \overline{\varphi}_i - \overline{\varphi}_j &\geq k'\beta_{ij} + \delta_{ij} \\
k'\beta_{ji} + \delta_{ji} \leq \overline{\varphi}_i - \overline{\varphi}_j &\leq (k'+1)\beta_{ji} - \delta_{ji}
\end{aligned}$$

■

Example 2.2.8: Returning to the set of patterns $\mathcal{P}^1$ considered in Example 2.2.6, Corollary 2.2.7 implies that $k' = -k - 1 = -2$. We verify that inequality (2.2.2) holds for $P_2(0.75)$ and $P_1(0)$.

$$\begin{aligned}
k'\beta_{21} + \delta_{21} &= -2(0.524) + 0.213 & (k'+1)\beta_{21} - \delta_{21} &= (-2+1)0.524 - 0.213 \\
&= -0.835 & &= -0.737
\end{aligned}$$

giving

$$\begin{aligned}
-0.835 \leq -0.75 &\leq -0.737 \\
k'\beta_{21} + \delta_{21} \leq \overline{\varphi}_1 - \overline{\varphi}_2 &\leq (k'+1)\beta_{21} - \delta_{21}
\end{aligned}$$

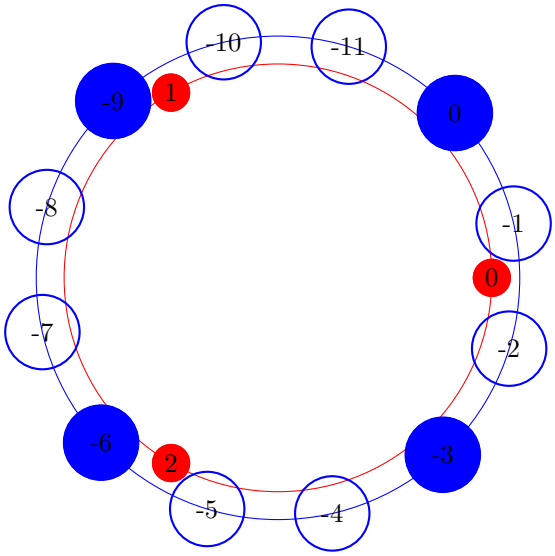


Figure 2.2.2: $\mathcal{P}^1$: The feasible template $\{P_2(0.75), P_1(0)\}$

Consider Figure 2.2.2 which contains the following:

- $P_2(0.75)$ shown as solid blue disks,
- $P_1(0)$ shown as solid red disks,
- $P_{21}(0.75)$ shown as both solid blue disks and open blue disks.

Both $P_1(0)$ and $P_{21}(0.75)$ are labelled with their indexing integers. This time note that since $k' = -2$, $P_1(0,0)$ lies between $P_{21}(0.75,-2)$ and $P_{21}(0.75,-1)$.

Theorem 2.2.4 gives a way of verifying that $\{\overline{\varphi}_i, \overline{\varphi}_j\} \in \mathcal{S}\big(\{P_i, P_j\}\big)$, but it requires finding the value of k first. The following corollary shows how this can be done.

Corollary 2.2.9 *Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given. If $\overline{\varphi}_i, \overline{\varphi}_j$ are valid for $\{P_i, P_j\}$ then*

$$k = \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor$$

satisfies inequality (2.2.2) *from Theorem 2.2.4.*

Proof: Assume that $P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) = \emptyset$. Thus, by Theorem 2.2.4, there exists $k \in \mathbb{Z}$ such that

$$\begin{gathered}
k\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - \delta_{ij} \\
\delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij} \leq \beta_{ij} - \delta_{ij} \\
\delta_{ij} - (\overline{\varphi}_j - \overline{\varphi}_i) \leq -k\beta_{ij} \leq \beta_{ij} - \delta_{ij} - (\overline{\varphi}_j - \overline{\varphi}_i) \\
\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij} - \beta_{ij} \leq k\beta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij} \\
\frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}} - 1 \leq k \leq \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}}.
\end{gathered}$$

As a result, we must have at least one integer inside the interval

$$\left[\frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}} - 1,\ \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}}\right] \tag{2.2.4}$$

and one such value is

$$k = \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor.$$

Additionally, if $\delta_{ij} > 0$,

$$\frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} < \frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}}.$$

From this it follows that interval (2.2.4) has length strictly less than 1 and thus contains a unique integer whose value is

$$k = \left\lceil \frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}} - 1 \right\rceil = \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor. \tag{2.2.5}$$

■

Example 2.2.10: Returning once again to the set $\mathcal{P}^1$ from Example 2.2.6, we calculate

$$\begin{aligned}
k &= \left\lfloor \frac{\overline{\varphi}_2 - \overline{\varphi}_1 - \delta_{12}}{\beta_{12}} \right\rfloor & k' &= \left\lfloor \frac{\overline{\varphi}_1 - \overline{\varphi}_2 - \delta_{21}}{\beta_{21}} \right\rfloor \\
&= \left\lfloor \frac{0.75 - 0 - 0.213}{0.524} \right\rfloor & &= \left\lfloor \frac{0 - 0.75 - 0.213}{0.524} \right\rfloor \\
&= \lfloor 1.025 \rfloor & &= \lfloor -1.838 \rfloor \\
&= 1 & &= -2.
\end{aligned}$$

Note that $k = -k' - 1$, which is consistent with Corollary 2.2.7.

The following corollary enables us to characterize $\mathcal{S}\big(\{P_i, P_j\}\big)$ without involving k.

Corollary 2.2.11 *Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given. Then $\overline{\varphi}_i, \overline{\varphi}_j$ are valid for $\{P_i, P_j\}$ iff*

$$\frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}} - 1 \leq \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor.$$

Proof: ($\Rightarrow$)

Assume that $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is a feasible template. Then equation (2.2.5) from the proof of Corollary 2.2.9 gives

$$k = \left\lceil \frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}} - 1 \right\rceil = \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor.$$

This implies that

$$\frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}} - 1 \leq \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor$$

as required.

($\Leftarrow$)

Assume that

$$\frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}} - 1 \leq \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor.$$

Let

$$k = \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor.$$

Then

$$\begin{aligned} k\beta_{ij} + \delta_{ij} &= \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor \beta_{ij} + \delta_{ij} \\ &\leq \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \cdot \beta_{ij} + \delta_{ij} \\ &= \overline{\varphi}_j - \overline{\varphi}_i. \end{aligned}$$

Also, by our assumption

$$\begin{aligned}(k+1)\beta_{ij} - \delta_{ij} &= \left(\left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}}{\beta_{ij}} \right\rfloor + 1 \right) \beta_{ij} - \delta_{ij} \\ &\geq \left(\frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}} - 1 + 1 \right) \beta_{ij} - \delta_{ij} \\ &= \frac{\overline{\varphi}_j - \overline{\varphi}_i + \delta_{ij}}{\beta_{ij}} \cdot \beta_{ij} - \delta_{ij} \\ &= \overline{\varphi}_j - \overline{\varphi}_i.\end{aligned}$$

So

$$k\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - \delta_{ij}$$

and thus k fulfills the requirements of Theorem 2.2.4. This implies that $P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) = \emptyset$. ■

Finally, the contents of $\mathcal{S}\big(\{P_i, P_j\}\big)$ can be verified by using the modulus function which we consider as the smallest non-negative remainder (i.e., $0 \leq a \bmod b < |b|$ for all $a, b \in \mathbb{R}$). The inequalities shown in the following two equivalent corollaries can be used to verify the validity of starting positions in a way that is neater and possibly simpler to calculate than by using Corollary 2.2.11.

Corollary 2.2.12 *Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given. Then $\overline{\varphi}_i, \overline{\varphi}_j$ are valid for $\{P_i, P_j\}$ iff*

$$\delta_{ij} \leq (\overline{\varphi}_j - \overline{\varphi}_i) \bmod \beta_{ij} \leq \beta_{ij} - \delta_{ij}.$$

Proof: Theorem 2.2.4 guarantees that $\overline{\varphi}_i, \overline{\varphi}_j$ are valid for $\{P_i, P_j\}$ iff there exists $k \in \mathbb{Z}$ such that

$$\begin{aligned}k\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - \delta_{ij} \\ \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij} \leq \beta_{ij} - \delta_{ij}\end{aligned}$$

which, since $\delta_{ij} \geq 0$, holds iff

$$\delta_{ij} \leq (\overline{\varphi}_j - \overline{\varphi}_i) \bmod \beta_{ij} \leq \beta_{ij} - \delta_{ij}.$$

■

We can equivalently express Corollary 2.2.12 as follows.

Corollary 2.2.13 *Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given. Then $\overline{\varphi}_i, \overline{\varphi}_j$ are valid for $\{P_i, P_j\}$ iff*

$$\big(\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}\big) \bmod \beta_{ij} \leq \beta_{ij} - 2\delta_{ij}. \tag{2.2.6}$$

Next, given two patterns, we can determine whether valid starting positions exist using the following corollary.

Corollary 2.2.14 *Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given. Then $\mathcal{S}\big(\{P_i, P_j\}\big) \neq \emptyset$ iff*

$$\beta_{ij} \geq 2\delta_{ij}.$$

Proof: ($\Rightarrow$)

Let $(\overline{\varphi}_i, \overline{\varphi}_j) \in \mathcal{S}\big(\{P_i, P_j\}\big)$ be given. Then, by Corollary 2.2.13 and the definition of the modulo function

$$0 \leq \big(\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}\big) \bmod \beta_{ij} \leq \beta_{ij} - 2\delta_{ij}$$

giving

$$\beta_{ij} \geq 2\delta_{ij}.$$

($\Leftarrow$)

Assume that $\beta_{ij} \geq 2\delta_{ij}$ or $0 \leq \beta_{ij} - 2\delta_{ij}$. If we let $\overline{\varphi}_i = 0$ and $\overline{\varphi}_j = \delta_{ij}$ then

$$\begin{aligned}\big(\overline{\varphi}_j - \overline{\varphi}_i - \delta_{ij}\big) \bmod \beta_{ij} &= (\delta_{ij} - 0 - \delta_{ij}) \bmod \beta_{ij} \\ &= 0 \\ &\leq \beta_{ij} - 2\delta_{ij}.\end{aligned}$$

Corollary 2.2.13 then shows that $(\overline{\varphi}_i, \overline{\varphi}_j) \in \mathcal{S}\big(\{P_i, P_j\}\big)$. ■

Example 2.2.15: Consider again the set $\mathcal{P}^1$ from Example 2.2.6. Starting with Corollary 2.2.14, we verify that

$$\beta_{12} = 0.524 \geq 0.426 = 2\delta_{12}$$

and so valid starting positions must exist for $\{P_1, P_2\}$. Next, let us verify that $\overline{\varphi}_1 = 0$ and $\overline{\varphi}_2 = 0.75$ are valid using the inequality from Corollary 2.2.13.

$$\begin{aligned}
(\overline{\varphi}_2 - \overline{\varphi}_1 - \delta_{12}) \bmod \beta_{12} &= (0.75 - 0 - 0.213) \bmod 0.524 \\
&= (0.537) \bmod 0.524 \\
&= 0.013 \\
&\leq 0.098 \\
&= 0.524 - 0.426 \\
&= \beta_{12} - 2\delta_{12}.
\end{aligned}$$

Finally, as the next corollary shows, if the two patterns have the same outer radius R, then the calculation of δ_{ij} becomes significantly simpler.

Corollary 2.2.16 *Let patterns $P_i = (R, n_i, \rho_i)$ and $P_j = (R, n_j, \rho_j)$ be given (i.e., $R_i = R_j = R$). Then $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is a feasible template iff there exists $k \in \mathbb{Z}$ such that*

$$k\beta_{ij} + 2\arcsin\left(\frac{\rho_i + \rho_j}{2R}\right) \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - 2\arcsin\left(\frac{\rho_i + \rho_j}{2R}\right). \tag{2.2.7}$$

Proof: Theorem 2.2.4 implies that $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is feasible iff there exists $k \in \mathbb{Z}$ such that

$$\begin{aligned}
k\beta_{ij} + 2\arcsin\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} &\leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - 2\arcsin\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
k\beta_{ij} + 2\arcsin\sqrt{\frac{(\rho_i + \rho_j)^2}{4R^2}} &\leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - 2\arcsin\sqrt{\frac{(\rho_i + \rho_j)^2}{4R^2}} \\
k\beta_{ij} + 2\arcsin\left(\frac{\rho_i + \rho_j}{2R}\right) &\leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - 2\arcsin\left(\frac{\rho_i + \rho_j}{2R}\right).
\end{aligned}$$

■

2.3 Multiple Patterns

We wish to extend the theory developed in the preceding section for two patterns to multiple patterns. This will enable us to solve the Feasible Fit Problem (FFP). Since a pattern is just a subset of $\mathbb{R}^2$, we can determine if a template is feasible by pairwise comparisons of the patterns in the template. This approach could be implemented in $O(|I|^2N^2)$ time, where N is the size of the largest n_i, but as this is pseudopolynomial it seems impractical. Recall that the inequality from Corollary 2.2.14 does not describe the φ_i's to be used; rather, it guarantees the existence of suitable φ_i values. To illustrate, assume we have three patterns $(R_1, n_1, \rho_1), (R_2, n_2, \rho_2), (R_3, n_3, \rho_3)$ where n_i, ρ_i and R_i pairwise satisfy the inequality from Corollary 2.2.14. Then we can find non-empty sets $\mathcal{S}\big(\{P_1, P_2\}\big)$, $\mathcal{S}\big(\{P_1, P_3\}\big)$ and $\mathcal{S}\big(\{P_2, P_3\}\big)$. The problem, however, is that the set

$$\mathcal{S}\big(\{P_1, P_2, P_3\}\big) = \Big\{(\varphi_1, \varphi_2, \varphi_3) \;\Big|\; (\varphi_1, \varphi_2) \in \mathcal{S}\big(\{P_1, P_2\}\big), (\varphi_1, \varphi_3) \in \mathcal{S}\big(\{P_1, P_3\}\big), (\varphi_2, \varphi_3) \in \mathcal{S}\big(\{P_2, P_3\}\big)\Big\}$$

may be empty. Rather, we need to find $\overline{\varphi}_1, \overline{\varphi}_2, \overline{\varphi}_3$ that satisfy inequality (2.2.6) from Corollary 2.2.13 for all possible pairs. In other words, it is not sufficient to ensure the existence of φ_i's pairwise; we need the φ_i's to be fixed first and then validated pairwise.

Before continuing, let us enforce a strict total ordering $\preceq$ on the patterns in the set. This will allow us to consider each pair only once. We now present the following theorem, whose proof follows directly from Theorem 2.2.4.

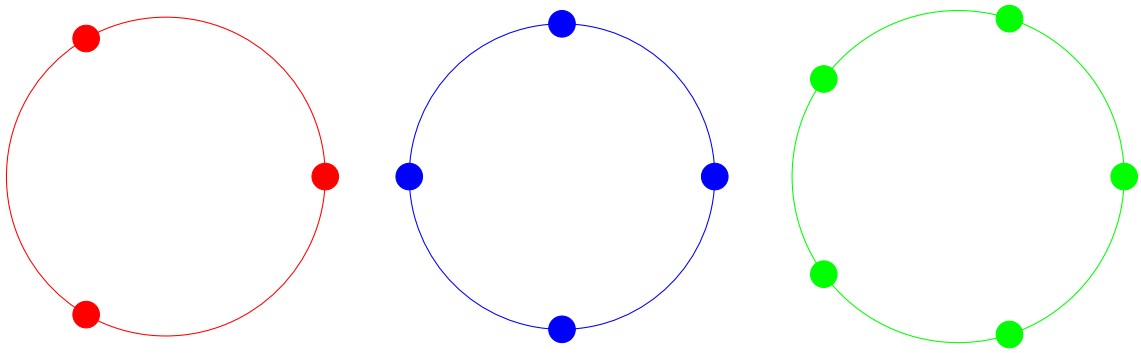
Theorem 2.3.1 *Suppose $\mathcal{P} = \{P_i \mid i \in I\}$ where $P_i = (R_i, n_i, \rho_i)$, and let $\preceq$ be a total order on I. Then $\{P_i(\overline{\varphi}_i) \mid i \in I\}$ is a feasible template iff there exist $k_{ij} \in \mathbb{Z}$, $i \prec j$, such that*

$$k_{ij}\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k_{ij}+1)\beta_{ij} - \delta_{ij}, \quad \forall i \prec j. \tag{2.3.1}$$

Thus

$$\mathcal{S}(\mathcal{P}) = \big\{\{\overline{\varphi}_i \mid i \in I\} \;\big|\; \exists k_{ij} \in \mathbb{Z} \ni k_{ij}\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k_{ij}+1)\beta_{ij} - \delta_{ij}, \quad \forall i \prec j\big\}. \tag{2.3.2}$$

Example 2.3.2: Consider the set $\mathcal{P}^2$ containing three patterns $P_1 = (12, 3, 1), P_2 = (11.5, 4, 1), P_3 = (12.5, 5, 1)$ shown in Figure 2.3.1. We use Corollary 2.2.13 to verify that $\{P_1(0), P_2(0.203), P_3(0.671)\}$ is a feasible template.



2.3.1(a): $P_1 = (12, 3, 1)$ 2.3.1(b): $P_2 = (11.5, 4, 1)$ 2.3.1(c): $P_3 = (12.5, 5, 1)$

Figure 2.3.1: $\mathcal{P}^2$: Fitting three patterns on a template

To begin, we calculate the following values:

$$\begin{aligned}
\beta_{12} &= \frac{2\pi}{[n_1, n_2]} = \frac{2\pi}{[3, 4]} \approx 0.524 \\
\beta_{13} &= \frac{2\pi}{[n_1, n_3]} = \frac{2\pi}{[3, 5]} \approx 0.419 \\
\beta_{23} &= \frac{2\pi}{[n_2, n_3]} = \frac{2\pi}{[4, 5]} \approx 0.314
\end{aligned}$$

$$\begin{aligned}
\delta_{12} &= 2\arcsin\sqrt{\frac{(\rho_1+\rho_2)^2-(R_1-R_2)^2}{4R_1R_2}} = 2\arcsin\sqrt{\frac{(1+1)^2-(12-11.5)^2}{4(12)(11.5)}} \\
&= 2\arcsin\sqrt{\frac{3.75}{552}} \approx 0.165 \\
\delta_{13} &= 2\arcsin\sqrt{\frac{(\rho_1+\rho_3)^2-(R_1-R_3)^2}{4R_1R_3}} = 2\arcsin\sqrt{\frac{(1+1)^2-(12-12.5)^2}{4(12)(12.5)}} \\
&= 2\arcsin\sqrt{\frac{3.75}{600}} \approx 0.158 \\
\delta_{23} &= 2\arcsin\sqrt{\frac{(\rho_2+\rho_3)^2-(R_2-R_3)^2}{4R_2R_3}} = 2\arcsin\sqrt{\frac{(1+1)^2-(11.5-12.5)^2}{4(11.5)(12.5)}} \\
&= 2\arcsin\sqrt{\frac{3}{575}} \approx 0.145.
\end{aligned}$$

We verify that inequality (2.2.6) holds for each pair i, j:

$$(\overline{\varphi}_2 - \overline{\varphi}_1 - \delta_{12}) \bmod \beta_{12} = (0.203 - 0 - 0.165) \bmod 0.524 = 0.038 \leq 0.194 = \beta_{12} - 2\delta_{12}$$

$$(\overline{\varphi}_3 - \overline{\varphi}_1 - \delta_{13}) \bmod \beta_{13} = (0.671 - 0 - 0.158) \bmod 0.419 = 0.094 \leq 0.103 = \beta_{13} - 2\delta_{13}$$

$$(\overline{\varphi}_3 - \overline{\varphi}_2 - \delta_{23}) \bmod \beta_{23} = (0.671 - 0.203 - 0.145) \bmod 0.314 = 0.009 \leq 0.024 = \beta_{23} - 2\delta_{23}.$$

Since inequality (2.2.6) is satisfied for all pairs i, j, the given starting positions must be valid. The resulting template is shown in Figure 2.3.2.

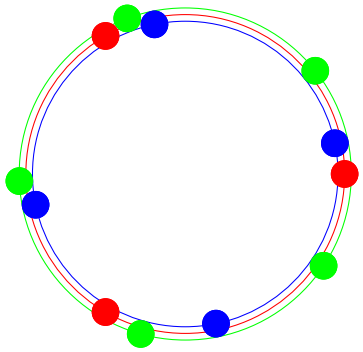


Figure 2.3.2: $\mathcal{P}^2$: The feasible template $\{P_1(0), P_2(0.203), P_3(0.671)\}$

2.3.1 Bounds on k_{ij}'s

Theorem 2.3.1 gives a characterization of valid starting positions for a set of patterns. However, a pattern contains rotational symmetry (as was detailed in Lemma 2.1.3) and this leads to many isomorphic solutions. For example $\{P_1(0), P_2(1)\}$ and $\{P_1(2\pi), P_2(1-2\pi)\}$ are geometrically the same template but have different sets of starting positions and therefore different k's. To reduce the number of candidate φ_i and k_{ij} values, we prove that, if valid starting positions exist, there must be a canonical set that lies in known intervals. Then we prove that the range of their associated k_{ij}'s is also limited. This will allow us to restrict attention to these specific, finite ranges.

Lemma 2.3.3 *Let $\mathcal{P} = \{P_i \mid i \in I\}$ be given. If $\{\overline{\varphi}_i \mid i \in I\} \in \mathcal{S}(\mathcal{P})$, then*

$$\left\{\hat{\varphi}_i = \overline{\varphi}_i \bmod \frac{2\pi}{n_i} \,\middle|\, i \in I\right\} \in \mathcal{S}(\mathcal{P}).$$

Proof: Let $\mathcal{P} = \{P_i \mid i \in I\}$ be given with $P_i = (R_i, n_i, \rho_i)$ and let $\{\overline{\varphi}_i \mid i \in I\} \in \mathcal{S}(\mathcal{P})$. Let $\hat{\varphi}_i = \overline{\varphi}_i \bmod \frac{2\pi}{n_i}$ for all $i \in I$. Then there exist $\kappa_i \in \mathbb{Z}$ such that

$$\hat{\varphi}_i = \overline{\varphi}_i + \kappa_i \frac{2\pi}{n_i}, \quad \forall i \in I$$

and $\hat{\varphi}_i \in \left[0, \frac{2\pi}{n_i}\right)$ for all $i \in I$. Let $p \neq q \in I$ be given. Then Property 2.1.3(b) gives

$$\begin{aligned} P_p(\hat{\varphi}_p) \cap P_q(\hat{\varphi}_q) &= P_p\left(\overline{\varphi}_p + \kappa_p \frac{2\pi}{n_p}\right) \cap P_q\left(\overline{\varphi}_q + \kappa_q \frac{2\pi}{n_q}\right) \\ &= P_p(\overline{\varphi}_p) \cap P_q(\overline{\varphi}_q), \end{aligned}$$

and since $\{\overline{\varphi}_i \mid i \in I\} \in \mathcal{S}(\mathcal{P})$, we obtain

$$\begin{aligned} P_p(\hat{\varphi}_p) \cap P_q(\hat{\varphi}_q) &= P_p(\overline{\varphi}_p) \cap P_q(\overline{\varphi}_q) \\ &= \emptyset. \end{aligned}$$

Thus $\{\hat{\varphi}_i \mid i \in I\} \in \mathcal{S}(\mathcal{P})$. ■

Now that we have a canonical range for the values of the φ's, we are able to bound the k_{ij} values similarly in the following theorem.

Theorem 2.3.4 *Let $\mathcal{P} = \{P_i \mid i \in I\}$ be given with $P_i = (R_i, n_i, \rho_i)$. Let $\preceq$ be a total order on the set I. If $\mathcal{S}(\mathcal{P}) \neq \emptyset$ then there exist*

$$k_{ij} \in \mathbb{Z} \cap \left[-\frac{n_j}{(n_i, n_j)},\ \frac{n_i}{(n_i, n_j)} - 1 \right], \quad \forall i \prec j$$

and $\hat{\varphi}_i \in \left[0, \dfrac{2\pi}{n_i}\right)$ for all $i \in I$ such that

$$k_{ij}\beta_{ij} + \delta_{ij} \leq \hat{\varphi}_j - \hat{\varphi}_i \leq (k_{ij} + 1)\beta_{ij} - \delta_{ij}, \quad \forall i \prec j.$$

Proof: Let $\mathcal{P} = \{P_i \mid i \in I\}$ be given with $P_i = (R_i, n_i, \rho_i)$ and assume that $\mathcal{T} = \{P_i(\overline{\varphi}_i) \mid i \in I\}$ is a feasible template. Define $\hat{\varphi}_i = \overline{\varphi}_i \bmod \dfrac{2\pi}{n_i}$ for all $i \in I$ which, by Lemma 2.3.3, are valid for $\mathcal{P}$. Finally, define $\preceq'$ to be a total order on I obtained by sorting the $\hat{\varphi}_i$'s. Namely, if

$$\hat{\varphi}_{\pi_1} \leq \hat{\varphi}_{\pi_2} \leq \hat{\varphi}_{\pi_3} \leq \cdots \leq \hat{\varphi}_{\pi_p}$$

then

$$\pi_1 \prec' \pi_2 \prec' \pi_3 \prec' \cdots \prec' \pi_p.$$

Now, since the $\hat{\varphi}_i$'s are valid starting positions, there must exist $\hat{k}_{ij} \in \mathbb{Z}$ such that

$$\hat{k}_{ij}\beta_{ij} + \delta_{ij} \leq \hat{\varphi}_j - \hat{\varphi}_i \leq (\hat{k}_{ij} + 1)\beta_{ij} - \delta_{ij}, \quad \forall i \prec' j. \tag{2.3.3}$$

Let $p \prec' q \in I$ be given and consider the left-hand side of inequality (2.3.3):

$$\begin{aligned} \hat{k}_{pq}\beta_{pq} + \delta_{pq} &\leq \hat{\varphi}_q - \hat{\varphi}_p \\ \hat{k}_{pq}\frac{2\pi}{[n_p, n_q]} &\leq \hat{\varphi}_q - \hat{\varphi}_p - \delta_{pq} \\ \hat{k}_{pq} &\leq (\hat{\varphi}_q - \hat{\varphi}_p - \delta_{pq})\frac{[n_p, n_q]}{2\pi}. \end{aligned}$$

Now $\hat{\varphi}_q - \hat{\varphi}_p - \delta_{pq} \leq \hat{\varphi}_q < \dfrac{2\pi}{n_q}$, thus

$$\begin{aligned}
\hat{k}_{pq} &< \frac{2\pi}{n_q} \cdot \frac{[n_p, n_q]}{2\pi} \\
\hat{k}_{pq} &< \frac{[n_p, n_q]}{n_q} \\
\hat{k}_{pq} &< \frac{n_p}{(n_p, n_q)} \\
\hat{k}_{pq} &\leq \frac{n_p}{(n_p, n_q)} - 1.
\end{aligned}$$

Using the right-hand side of (2.3.3), we get

$$\begin{aligned}
\hat{\varphi}_q - \hat{\varphi}_p &\leq (\hat{k}_{pq} + 1)\beta_{pq} - \delta_{pq} \\
\hat{\varphi}_q - \hat{\varphi}_p + \delta_{pq} &\leq (\hat{k}_{pq} + 1)\frac{2\pi}{[n_p, n_q]} \\
(\hat{\varphi}_q - \hat{\varphi}_p + \delta_{pq})\frac{[n_p, n_q]}{2\pi} &\leq \hat{k}_{pq} + 1.
\end{aligned}$$

This time, since $\hat{\varphi}_q \geq \hat{\varphi}_p$, we get $\hat{\varphi}_q - \hat{\varphi}_p + \delta_{pq} \geq \delta_{pq} \geq 0$. Thus

$$\begin{aligned}
0 &\leq \hat{k}_{pq} + 1 \\
-1 &\leq \hat{k}_{pq}
\end{aligned}$$

Since the choice of p and q was arbitrary, we have $\hat{k}_{ij} \in \left[-1, \dfrac{n_i}{(n_i, n_j)} - 1\right]$ for all $i \prec' j$. Now let $\preceq$ be any total order on I and for $i \prec j$ define

$$\overline{k}_{ij} = \begin{cases} \hat{k}_{ij}, & i \prec' j \\ -\hat{k}_{ji} - 1, & j \prec' i. \end{cases}$$

It follows from (2.3.3) and Theorem 2.2.7 that

$$\overline{k}_{ij}\beta_{ij} + \delta_{ij} \leq \hat{\varphi}_j - \hat{\varphi}_i \leq (\overline{k}_{ij} + 1)\beta_{ij} - \delta_{ij}, \quad \forall i \prec j$$

so all that remains is to show that

$$-\frac{n_j}{(n_i, n_j)} \leq \overline{k}_{ij} \leq \frac{n_i}{(n_i, n_j)} - 1, \quad \forall i \prec j.$$

Let $p \prec q$ be given. If $p \prec' q$ then

$$\overline{k}_{pq} = \hat{k}_{pq} \in \left[-1, \frac{n_p}{(n_p, n_q)} - 1\right]$$

else, if $q \prec' p$ then

$$\begin{aligned} -1 \leq \hat{k}_{qp} &\leq \frac{n_q}{(n_q, n_p)} - 1 \\ 0 \leq \hat{k}_{qp} + 1 &\leq \frac{n_q}{(n_q, n_p)} \\ -\frac{n_q}{(n_q, n_p)} \leq -\hat{k}_{qp} - 1 &\leq 0 \\ -\frac{n_q}{(n_p, n_q)} \leq \overline{k}_{pq} &\leq 0. \end{aligned}$$

Thus, in either case $\overline{k}_{pq} \in \left[-\frac{n_q}{(n_p, n_q)}, \frac{n_p}{(n_p, n_q)} - 1\right]$. Since our selection of p and q was arbitrary we have

$$-\frac{n_j}{(n_i, n_j)} \leq \overline{k}_{ij} \leq \frac{n_i}{(n_i, n_j)} - 1, \quad \forall i \prec j$$

as required. ■

2.3.2 Integer Programming Formulations

Theorem 2.3.1 identifies a set of constraints involving both integer and real variables. This situation lends itself to the application of integer programming [29]. The Feasible Fit Problem (FFP) can be

formulated as a mixed integer program (MIP) with no objective function:

$$\begin{aligned}
\textsc{Fit}(\mathcal{P}): \qquad & k_{ij}\beta_{ij} + \varphi_i - \varphi_j \leq -\delta_{ij} && \forall i \prec j && (2.3.4)\\
& \varphi_j - \varphi_i - k_{ij}\beta_{ij} \leq \beta_{ij} - \delta_{ij} && \forall i \prec j && (2.3.5)\\
& 0 \leq \varphi_i \leq \frac{2\pi}{n_i} && \forall i \in I && (2.3.6)\\
& -\frac{n_j}{(n_i, n_j)} \leq k_{ij} \leq \frac{n_i}{(n_i, n_j)} - 1 && \forall i \prec j && (2.3.7)\\
& k_{ij} \in \mathbb{Z} && \forall i \prec j\\
& \varphi_i \in \mathbb{R} && \forall i \in I.
\end{aligned}$$

In $\textsc{Fit}(\mathcal{P})$, constraints (2.3.4) and (2.3.5) are the two sides of inequality (2.3.1) from Theorem 2.3.1, whereas constraints (2.3.6) and (2.3.7) are obtained from Theorem 2.3.4.

In the Maximum Template Problem (MaxTemp) we are given a set of patterns $\mathcal{P} = \{P_i \mid i \in I\}$ and we wish to build a largest feasible template $\mathcal{T} = \{P_i(\varphi_i) \mid i \in I_1 \subseteq I\}$. To formulate this as a mixed integer program we introduce binary variables x_i that determine whether or not pattern P_i is in the maximum template $\mathcal{T}$: namely, for $i \in I$

$$x_i = \begin{cases} 1, & i \in I_1 \\ 0, & i \notin I_1. \end{cases}$$

Recall that inequality (2.3.1) ensures that two patterns $P_i(\varphi_i)$ and $P_j(\varphi_j)$ do not overlap by insisting that the difference in their starting positions lies in exactly one of a set of non-overlapping intervals

$$\Big\{ \big[k_{ij}\beta_{ij} + \delta_{ij},\ (k_{ij}+1)\beta_{ij} - \delta_{ij}\big] \ \Big|\ k_{ij} \in \mathbb{Z} \Big\}. \tag{2.3.8}$$

However, if P_i and P_j are not both on the maximum template $\mathcal{T}$, we do not want to constrain the difference in their starting positions. In other words, we do not care about enforcing inequality (2.3.1) for two patterns that are not on the same template. We can do this by modifying the set of intervals in (2.3.8) so that their union is the entire set $\mathbb{R}$. In that case there will be an interval

containing $\overline{\varphi}_j - \overline{\varphi}_i$ no matter what their values. To achieve this, we change the intervals (2.3.8) to

$$\Big\{ \big[k_{ij}\beta_{ij}+\delta_{ij}-(1-x_i)\delta_{ij}-(1-x_j)\delta_{ij},\ (k_{ij}+1)\beta_{ij}-\delta_{ij}+(1-x_i)\delta_{ij}+(1-x_j)\delta_{ij}\big] \ \Big|\ k_{ij} \in \mathbb{Z} \Big\}. \quad (2.3.9)$$

Notice that if x_i and x_j both have value one, then the set of intervals (2.3.9) is identical to (2.3.8). However if either or both variables have value zero, then the union of the intervals in (2.3.9) is $\mathbb{R}$ as required. The MaxTemp problem can now be formulated as

$$\begin{aligned}
&\textsc{MaxTemplate}(\mathcal{P}, \preceq) : \max \sum_{i \in I} x_i \\
\text{s.t. } & k_{ij}\beta_{ij} + \varphi_i - \varphi_j + x_i\delta_{ij} + x_j\delta_{ij} \leq \delta_{ij} && \forall i \prec j && (2.3.10)\\
& \varphi_j - \varphi_i - k_{ij}\beta_{ij} + x_i\delta_{ij} + x_j\delta_{ij} \leq \beta_{ij} + \delta_{ij} && \forall i \prec j && (2.3.11)\\
& 0 \leq \varphi_i \leq \frac{2\pi}{n_i} && \forall i \in I \\
& -\frac{n_j}{(n_i, n_j)} \leq k_{ij} \leq \frac{n_i}{(n_i, n_j)} - 1 && \forall i \prec j \\
& k_{ij} \in \mathbb{Z} && \forall i \prec j \\
& \varphi_i \in \mathbb{R} && \forall i \in I \\
& x_i \in \{0, 1\} && \forall i \in I.
\end{aligned}$$

In $\textsc{MaxTemplate}(\mathcal{P}, \preceq)$, the objective function maximizes the number of patterns in the template, which is equal to the sum of the x variables. Also, constraints (2.3.10) and (2.3.11) implement the inequality from Theorem 2.3.1 with the modifications in (2.3.9). The next two sets of constraints are repeated from $\textsc{Fit}(\mathcal{P}, \preceq)$.

In the Minimum Templates Problem (MinTemp) we are given a set $\mathcal{P} = \{P_i \mid i \in I\}$ and we wish to partition I into disjoint subsets $\{I_u \mid u \in U\}$. These subsets need to be such that $\mathcal{T}_u = \{P_i(\varphi_i) \mid i \in I_u\}$ is a feasible template for all $u \in U$. To formulate this as a mixed integer program define binary variables x to indicate which template each pattern is assigned to:

$$x_{iu} = \begin{cases} 1, & i \in I_u \\ 0, & i \notin I_u \end{cases}, \quad \forall i \in I, \forall u \in U.$$

We also duplicate the constraint set from MaxTemplate$(\mathcal{P}, \preceq)$ for each template. Because of this duplication, it will significantly reduce the size of the MIP if we have a good upper bound[2] on the number of templates needed, $|U|$. Additionally, we define binary variables z_u to indicate whether or not a template is empty:

$$z_u = \begin{cases} 1, & I_u \neq \emptyset \\ 0, & I_u = \emptyset \end{cases}, \quad \forall u \in U.$$

The resulting formulation[3] is

$$\begin{aligned}
&\text{MinTemplates}(\mathcal{P}, \preceq, U) : \min \sum_{u \in U} z_u \\
&\text{s.t. } k_{ij}\beta_{ij} + \varphi_i - \varphi_j + x_{iu}\delta_{ij} + x_{ju}\delta_{ij} \leq \delta_{ij} && \forall i \prec j, u \in U && (2.3.12)\\
&\varphi_j - \varphi_i - k_{ij}\beta_{ij} + x_{iu}\delta_{ij} + x_{ju}\delta_{ij} \leq \beta_{ij} + \delta_{ij} && \forall i \prec j, u \in U && (2.3.13)\\
&\sum_{u \in U} x_{iu} = 1 && \forall i \in I && (2.3.14)\\
&x_{iu} - z_u \leq 0 && \forall i \in I, u \in U && (2.3.15)\\
&0 \leq \varphi_i \leq \frac{2\pi}{n_i} && \forall i \in I \\
&-\frac{n_j}{(n_i, n_j)} \leq k_{ij} \leq \frac{n_i}{(n_i, n_j)} - 1 && \forall i \prec j \\
&k_{ij} \in \mathbb{Z} && \forall i \prec j \\
&\varphi_i \in \mathbb{R} && \forall i \in I \\
&x_{iu} \in \{0, 1\} && \forall i \in I, u \in U \\
&z_u \geq 0 && \forall u \in U.
\end{aligned}$$

In this MIP, the objective function minimizes the number of non-empty templates, given by the sum of the z variables. Constraints (2.3.12)-(2.3.13) are modified versions of constraints (2.3.10)-(2.3.11), duplicated for each template. Constraint (2.3.14) ensures that each pattern is assigned to exactly one template. If a pattern P_i is assigned to template $\mathcal{T}_u$ then $x_{iu} = 1$ and constraint (2.3.15) forces $z_u = 1$, ensuring that template $\mathcal{T}_u$ is non-empty. The next two sets of constraints are repeated from MaxTemplate$(\mathcal{P}, \prec)$. Now observe that the objective function involves minimizing variables z_u;

[2]This bound can be generated by suitably designed heuristics such as those discussed in Chapter 5, or it can naïvely be set to $|I|$.

[3]This formulation is based on that developed by Eisenbrand, et al. [11].

moreover z_u only occurs in the constraints (2.3.15). Because of the structure of this constraint, if $z_u > 0$ then $z_u = 1$ in any optimal solution. Thus we relax the binary condition on z_u, allowing it to be continuous and non-negative.

Integer programming is a very versatile tool for solving a myriad of problems and there is a vast array of software that will attempt to solve any integer program to optimality. Unfortunately, however, as the number of discrete variables in a mixed integer program grows, known techniques for solving it become ineffective. This is especially evident in the mixed integer programs presented in this section as the number of feasible integer points grows rapidly in relation to the number of patterns in $\mathcal{P}$. Table 2.3.1 summarizes the sizes of these formulations.

Formulation	FIT($\mathcal{P}$)	MAXTEMPLATE($\mathcal{P}$)	MINTEMPLATES($\mathcal{P}$)
Constraints	$\lvert I\rvert\big(\lvert I\rvert-1\big)$	$\lvert I\rvert\big(\lvert I\rvert-1\big)$	$\lvert U\rvert\lvert I\rvert^2+\lvert I\rvert$
Real Variables	$\lvert I\rvert$	$\lvert I\rvert$	$\lvert I\rvert+\lvert U\rvert$
Integer Variables	$\frac{\lvert I\rvert\big(\lvert I\rvert-1\big)}{2}$	$\frac{\lvert I\rvert\big(\lvert I\rvert-1\big)}{2}+\lvert I\rvert$	$\frac{\lvert I\rvert\big(\lvert I\rvert-1\big)}{2}+\lvert U\rvert\lvert I\rvert$

Table 2.3.1: Mixed integer programming formulation sizes

2.4 Network Representation

Consider inequalities (2.3.1) for any pair of patterns P_i and P_j where $i \prec j$:

$$k_{ij}\beta_{ij} + \delta_{ij} \leq \varphi_j - \varphi_i \leq (k_{ij}+1)\beta_{ij} - \delta_{ij}$$

$$\begin{cases} \varphi_j - \varphi_i \leq (k_{ij}+1)\beta_{ij} - \delta_{ij} \\ \varphi_i - \varphi_j \leq -k_{ij}\beta_{ij} - \delta_{ij}. \end{cases}$$

Thus, if we define for $i \neq j$

$$w_{ij} = \begin{cases} (k_{ij}+1)\beta_{ij} - \delta_{ij}, & i \prec j \\ -k_{ji}\beta_{ji} - \delta_{ji}, & j \prec i \end{cases}$$

then the above inequalities become (for $i \neq j$)

$$\varphi_j - \varphi_i \leq w_{ij}$$

or

$$\varphi_j \leq \varphi_i + w_{ij}. \tag{2.4.1}$$

These, however, are simply the dual program constraints for the single-source shortest path problem [1] in an appropriately defined network with arc lengths (weights) w_{ij}. Thus, any feasible dual variables will be valid starting positions. Additionally, if we specify a root node (i.e., $\varphi_p = 0$), then the shortest path distances from the root node will be valid starting positions. Consequently, we are led to the construction of the following network.

Definition 2.4.1 *Let $\mathcal{P} = \{P_i \mid i \in I\}$ be a set of patterns where $P_i = (R_i, n_i, \rho_i)$, let $\preceq$ be an order on I, and let $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$. We define the network $G(\mathcal{P}, \preceq, K) = (N, A, w)$ where*

$$N = I$$

$$A = \big\{(i,j) \;\big|\; i \neq j \in I\big\}$$

$$w_{ij} = \begin{cases} (k_{ij}+1)\beta_{ij} - \delta_{ij}, & i \prec j \\ -k_{ji}\beta_{ji} - \delta_{ji}, & j \prec i. \end{cases}$$

The following theorem formalizes the connection between shortest paths in $G(\mathcal{P}, \preceq, K)$ from a given source node and valid starting positions.

Theorem 2.4.2 *Let $\mathcal{P} = \{P_i \mid i \in I\}$ be a set of patterns, let $\preceq$ be an order on I, and let $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$. If $\{\overline{\varphi}_i \mid i \in I\}$ are shortest path distances from a specified root node in network $G(\mathcal{P}, \preceq, K)$, then $\{\overline{\varphi}_i \mid i \in I\} \in \mathcal{S}(\mathcal{P})$.*

Proof: Since the shortest path distances from the specified root satisfy the shortest path optimality

conditions, then for any $i \prec j$

$$\begin{aligned} \overline{\varphi}_i - \overline{\varphi}_j &\leq w_{ji} & \overline{\varphi}_j - \overline{\varphi}_i &\leq w_{ij} \\ \overline{\varphi}_i - \overline{\varphi}_j &\leq -k_{ij}\beta_{ij} - \delta_{ij} & \overline{\varphi}_j - \overline{\varphi}_i &\leq (k_{ij}+1)\beta_{ij} - \delta_{ij} \\ k_{ij}\beta_{ij} + \delta_{ij} &\leq \overline{\varphi}_j - \overline{\varphi}_i & \overline{\varphi}_j - \overline{\varphi}_i &\leq (k_{ij}+1)\beta_{ij} - \delta_{ij}. \end{aligned}$$

Thus the $\overline{\varphi}_i$'s satisfy characterization (2.3.2) for $\mathcal{S}(\mathcal{P})$. ■

Similarly, one can find shortest paths in $G(\mathcal{P}, \preceq, K)$ to a specified root node. The associated optimality conditions lead to the following corollary.

Corollary 2.4.3 *Let $\mathcal{P} = \{P_i \mid i \in I\}$ be a set of patterns, let $\preceq$ be an order on I, and let $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$. If $\left\{\overline{\psi}_i \;\middle|\; i \in I\right\}$ are valid shortest path distances to a specified root node in network $G(\mathcal{P}, \preceq, K)$, then $\left\{-\overline{\psi}_i \;\middle|\; i \in I\right\} \in \mathcal{S}(\mathcal{P})$.*

Proof: The optimality conditions for shortest paths to a given root node are

$$\begin{aligned} \overline{\psi}_i &\leq w_{ij} + \overline{\psi}_j \\ \overline{\psi}_i - \overline{\psi}_j &\leq w_{ij} \\ -(-\overline{\psi}_i) + (-\overline{\psi}_j) &\leq w_{ij} \\ (-\overline{\psi}_j) - (-\overline{\psi}_i) &\leq w_{ij}. \end{aligned}$$

Now letting $\overline{\varphi}_i = -\overline{\psi}_i$ for all $i \in I$, we get

$$\overline{\varphi}_j - \overline{\varphi}_i \leq w_{ij}.$$

Thus, by the same argument as above, the $\overline{\varphi}_i$'s or $\left(-\overline{\psi}_i\right)$'s satisfy characterization (2.3.2) for $\mathcal{S}(\mathcal{P})$. ■

Example 2.4.4: Recall the set $\mathcal{P}^2$ from Example 2.3.2 which contains patterns $P_1 = (12, 3, 1), P_2 = (11.5, 4, 1), P_3 = (12.5, 5, 1)$. If we use the same ordering $1 \prec 2 \prec 3$ and treat K as unknown, then the network $G(\mathcal{P}, \preceq, K)$ is shown in Figure 2.4.1.

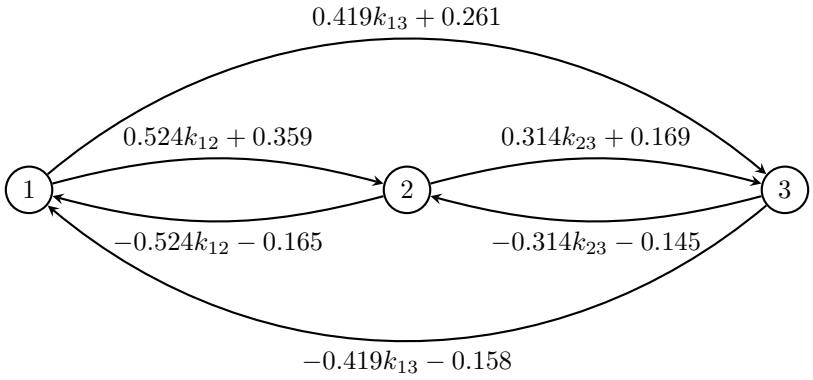


Figure 2.4.1: $\mathcal{P}^2$: $G(\mathcal{P}, \preceq, K)$

If we propose $\overline{K} = \{\overline{k}_{12} = 0, \overline{k}_{13} = 1, \overline{k}_{23} = 1\}$ then $G(\mathcal{P}, \preceq, \overline{K})$ is shown in Figure 2.4.2.

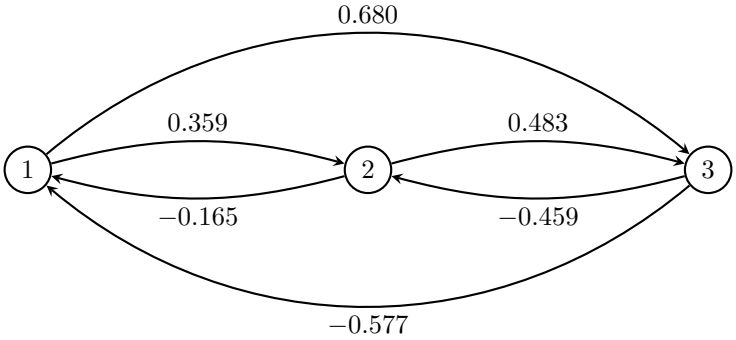
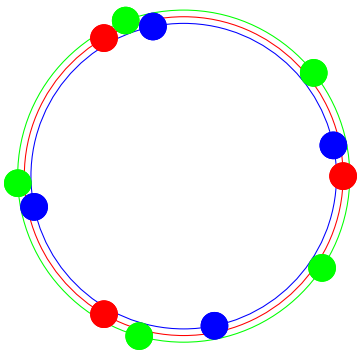
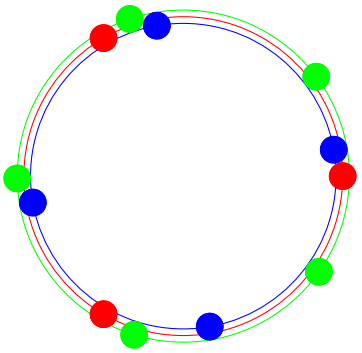


Figure 2.4.2: $\mathcal{P}^2$: $G(\mathcal{P}, \preceq, \overline{K})$

Finding shortest paths from node 1 gives (0, 0.221, 0.680) = $(\overline{\varphi}_1, \overline{\varphi}_2, \overline{\varphi}_3)$. This solution is shown in Figure 2.4.3(a). Alternatively, finding shortest paths to node 1 gives (0, −0.165, −0.624) = $(-\overline{\varphi}_1, -\overline{\varphi}_2, -\overline{\varphi}_3)$. This solution is shown in Figure 2.4.3(b).



2.4.3(a): The feasible template
$\{P_1(0), P_2(0.221), P_3(0.680)\}$



2.4.3(b): The feasible template
$\{P_1(0), P_2(0.165), P_3(0.624)\}$

Figure 2.4.3: $\mathcal{P}^2$: Solutions gained from the network

Now we know that if we can find shortest paths in network $G(\mathcal{P}, \preceq, K)$, then the corresponding shortest path distances are valid starting positions for $\mathcal{P}$. However, shortest path distances exist in a network iff that network contains no negative weight cycle. This leads to the following theorem.

Theorem 2.4.5 *Let $\mathcal{P} = \{P_i \mid i \in I\}$ be a set of patterns and let $\preceq$ be an order on I. $\mathcal{S}(\mathcal{P}) \neq \emptyset$ iff there exists $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$ such that $G(\mathcal{P}, \preceq, K)$ contains no negative (weight) cycle.*

Proof:

($\Leftarrow$)

Since $G(\mathcal{P}, \preceq, K)$ has no negative cycles, a set of shortest path distances from any node in I will exist [1] and by Theorem 2.4.2 will be an element of $\mathcal{S}(\mathcal{P})$.

($\Rightarrow$)

Let $\{\overline{\varphi}_i \mid i \in I\} \in \mathcal{S}(\mathcal{P})$ be given, and let $\preceq$ be an order on I. By Theorem 2.3.1 there is a set $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$ satisfying

$$k_{ij}\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k_{ij} + 1)\beta_{ij} - \delta_{ij}, \quad \forall i \prec j$$

or

$$\overline{\varphi}_j - \overline{\varphi}_i \leq (k_{ij} + 1)\beta_{ij} - \delta_{ij}, \quad \forall i \prec j$$

and

$$\overline{\varphi}_j - \overline{\varphi}_i \leq -k_{ji}\beta_{ji} - \delta_{ji}, \quad \forall j \prec i.$$

Let C be any directed cycle in $G(\mathcal{P}, \preceq, K)$ and define the following sets

$$\begin{aligned} C_{\prec} &:= \{(i,j) \in C \mid i \prec j\} \\ C_{\succ} &:= \{(i,j) \in C \mid j \prec i\}. \end{aligned} \tag{2.4.2}$$

Now, the total weight of the cycle C is given by

$$\begin{aligned} w[C] &= \sum_{(i,j)\in C} w_{ij} \\ &= \sum_{(i,j)\in C_{\prec}} [(k_{ij}+1)\beta_{ij} - \delta_{ij}] + \sum_{(i,j)\in C_{\succ}} [-k_{ji}\beta_{ji} - \delta_{ji}] \\ &\geq \sum_{(i,j)\in C_{\prec}} (\overline{\varphi}_j - \overline{\varphi}_i) + \sum_{(i,j)\in C_{\succ}} (\overline{\varphi}_j - \overline{\varphi}_i) \\ &= \sum_{(i,j)\in C} (\overline{\varphi}_j - \overline{\varphi}_i) \\ &= 0. \end{aligned}$$

Thus the weight of cycle C must be non-negative. Since the choice of C was arbitrary, $G(\mathcal{P}, \preceq, K)$ must have no negative cycle. ■

We have already seen that the existence of starting positions leading to a feasible template is not dependent on the order we assign to the patterns. However, in our network representation of the problem we have imposed an ordering, which affects the calculation of the w_{ij}'s. The following theorem shows that if we change the ordering $\preceq$, there is a corresponding change to the set K that will keep the network invariant.

Theorem 2.4.6 *Let $\mathcal{P} = \{P_i \mid i \in I\}$ be a set of patterns, let $\preceq$ be a total order on I, and let $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$. Let $p \prec q$ be such that there does not exist $r \in I$ satisfying $p \prec r \prec q$ (i.e., p and q are adjacent w.r.t. $\preceq$). Define $\preceq'$ by $q \prec' p$ and*

$$i \prec' j \Leftrightarrow i \prec j, \quad \forall \{i,j\} \neq \{p,q\} \subseteq I$$

i.e., $\preceq'$ is exactly the same as $\preceq$ except that the order of p and q is switched. Also, let $K' = \{k'_{ij} \mid i \prec' j\}$ where

$$k'_{ij} = \begin{cases} -k_{ji} - 1, & i = q,\ j = p \\ k_{ij}, & \text{else.} \end{cases}$$

Then $G(\mathcal{P}, \preceq, K) = G(\mathcal{P}, \preceq', K')$.

Proof: Both networks have the same node set, and since both networks are complete the only question concerns the equality of the weights. Let w denote the weights for $G(\mathcal{P}, \preceq, K)$ and let w' denote the weights for $G(\mathcal{P}, \preceq', K')$. Consider $\{i,j\} \subseteq I$. If $\{i,j\} \neq \{p,q\}$ then $w_{ij} = w'_{ij}$. Now $q \prec' p$ and $p \prec q$, so

$$\begin{aligned} w'_{pq} &= -k'_{qp}\beta_{qp} - \delta_{qp} \\ &= -(-k_{pq} - 1)\beta_{qp} - \delta_{pq} \\ &= (k_{pq} + 1)\beta_{pq} - \delta_{pq} \\ &= w_{pq} \end{aligned} \qquad \begin{aligned} w'_{qp} &= (k'_{qp} + 1)\beta_{qp} - \delta_{qp} \\ &= (-k_{pq} - 1 + 1)\beta_{qp} - \delta_{qp} \\ &= -k_{pq}\beta_{pq} - \delta_{pq} \\ &= w_{qp}. \end{aligned}$$

Thus $w = w'$ and the two networks are identical. ■

Theorem 2.4.6 shows that given network $G(\mathcal{P}, \preceq, K)$, we will obtain an identical network $G(\mathcal{P}, \preceq', K')$ where $\preceq$ and $\preceq'$ have one pair of adjacent elements reversed. Since any permutation can be obtained from any other permutation through a sequence of pairwise swaps, Theorem 2.4.6 ensures that given network $G(\mathcal{P}, \preceq, K)$ and any other total order $\preceq'$ on I, there exists K' such that

$$G(\mathcal{P}, \preceq, K) = G(\mathcal{P}, \preceq', K').$$

The construction in Theorem 2.4.6 can be applied to the given K repeatedly to obtain the corre-

sponding set K'.

The next theorem investigates the special case of inverse orderings.

Theorem 2.4.7 *Let $\mathcal{P} = \{P_i \mid i \in I\}$ be a set of patterns. Let $\preceq_1$ and $\preceq_2$ be total orderings on I such that*

$$i \preceq_1 j \Leftrightarrow j \preceq_2 i, \quad \forall i, j \in I,$$

i.e., $\preceq_1$ and $\preceq_2$ are inverses of each other. Let $K^1 = \{k^1_{ij} \in \mathbb{Z} \mid i \prec_1 j\}$ be given and let $K^2 = \{k^2_{ij} \in \mathbb{Z} \mid i \prec_2 j\}$ be defined by $k^2_{ij} = k^1_{ji}$ for all $i \prec_2 j$. Then $G(\mathcal{P}, \preceq_1, K^1)$ has no negative weight cycle iff $G(\mathcal{P}, \preceq_2, K^2)$ has no negative weight cycle.

Proof: Assume that $G(\mathcal{P}, \preceq_1, K^1)$ has no negative weight cycle. Let cycle $C \in G(\mathcal{P}, \preceq_2, K^2)$ be given. Note that $C \in G(\mathcal{P}, \preceq_1, K^1)$ since both networks are complete. Define $\overline{C}$ to be the reverse cycle of C: $\overline{C} = \{(i,j) \mid (j,i) \in C\}$. Let $C^1_\prec, C^1_\succ, \overline{C}^1_\prec \overline{C}^1_\succ$ be defined relative to $\preceq_1$ as in (2.4.2) and similarly define $C^2_\prec, C^2_\succ, \overline{C}^2_\prec \overline{C}^2_\succ$ relative to $\preceq_2$. First note that

$$\begin{aligned}(i,j) \in C^2_\prec &\Leftrightarrow i \prec_2 j \\ &\Leftrightarrow j \prec_1 i \\ &\Leftrightarrow (j,i) \in \overline{C}^1_\prec\end{aligned}$$

and similarly $(i,j) \in C^2_\succ \Rightarrow (j,i) \in \overline{C}^1_\succ$. Now, the total weight of C in $G(\mathcal{P}, \preceq_2, K^2)$ is given by

$$\begin{aligned}w^2[C] &= \sum_{(i,j)\in C^2_\prec} \left[(k^2_{ij}+1)\beta_{ij} - \delta_{ij}\right] + \sum_{(i,j)\in C^2_\succ} \left[-k^2_{ji}\beta_{ji} - \delta_{ji}\right] \\ &= \sum_{(j,i)\in \overline{C}^1_\prec} \left[(k^1_{ji}+1)\beta_{ji} - \delta_{ji}\right] + \sum_{(j,i)\in \overline{C}^1_\succ} \left[-k^1_{ij}\beta_{ij} - \delta_{ij}\right] \\ &= w^1[\overline{C}] \\ &\geq 0.\end{aligned}$$

Thus $G(\mathcal{P}, \preceq_2, K^2)$ also has no negative weight cycle. ■

In summary, if we can find an appropriate set K, then we can calculate valid starting positions

quickly and easily using well-known shortest path algorithms. The problem of finding valid starting positions is now reduced to the problem of finding K such that $G(\mathcal{P}, \preceq, K)$ has no negative cycles. Heuristics that address this approach will be covered in Chapter 5, but for now we continue with a relaxation of the original problem.

Chapter 3

Linearization

So far we have dealt with an exact formulation of the problem of fitting patterns without overlap. However, this formulation involves the calculation of irrational constants β_{ij} and δ_{ij} which leads to some computational challenges. In this chapter we wish to simplify the problem by eliminating a dimension of the problem, i.e., making a pattern a subset of $\mathbb{R}$ instead of $\mathbb{R}^2$. In the process, we make a connection between the problem of fitting patterns of disks and the problem of scheduling periodic events. We then show how the results of Chapter 2 can be extended to this simplified version of the problem and consider some special cases that are exploited by heuristics developed in Chapter 5.

3.1 Definitions

The structure of a pattern can be approximated by unfolding its outer circumference along the number line and duplicating the pattern infinitely in both directions. The disks of the pattern now become intervals of the outer circumference. The pattern can now be defined by the length of the outer circumference B, the number of intervals in the pattern n, and the length of each interval d. The rotation of the pattern now becomes a shift s, which will measure the distance from a designated point on the outer circumference to the left endpoint of the first interval.

Definition 3.1.1 *A linear pattern maps a starting position s to an infinite set of open intervals: i.e.,*

$$\widetilde{P}(s) = \bigcup_{k\in\mathbb{Z}} \Big(s + k\frac{B}{n},\ \ s + k\frac{B}{n} + d \Big).$$

The mapping is completely defined by the strictly positive constants $B, d \in \mathbb{R}$ and $n \in \mathbb{Z}$ and thus can also be designated as $\widetilde{P} = (B, n, d)$.

Definition 3.1.2 *A template is a set of linear patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ with fixed starting positions $S = \{\overline{s}_i \mid i \in I\}$ and will be denoted $\mathcal{T} = \widetilde{\mathcal{P}}(S) = \{\widetilde{P}_i(\overline{s}_i) \mid i \in I\}$. Furthermore, if*

$$\widetilde{P}_i(\overline{s}_i) \cap \widetilde{P}_j(\overline{s}_j) = \emptyset, \quad \forall i \neq j$$

then $\widetilde{\mathcal{T}}$ is referred to as a feasible template and the starting positions S are said to be valid for $\widetilde{\mathcal{P}}$. The collection of all starting positions for which $\widetilde{\mathcal{P}}$ is a feasible template is defined as

$$\mathcal{S}(\widetilde{\mathcal{P}}) = \big\{ S \,\big|\, \widetilde{\mathcal{P}}(S) \text{ is a feasible template} \big\}.$$

We further simplify our patterns by requiring that all patterns in a set share the same outer circumference B. In this simplified context, we study three questions analogous to the ones defined in Chapter 2: namely, the Linearized Minimum Templates Problem (LinMinTemp), the Linearized Feasible Fit Problem (LinFFP) and the Linearized Maximum Template Problem (LinMaxTemp).

LinMinTemp($\widetilde{\mathcal{P}}$)**:**
Given a set of linear patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ where $\widetilde{P}_i = (B, n_i, d_i)$, determine a partition of I, $\mathcal{I} = \{I_u \mid u \in U\}$, with minimum $\lvert U\rvert$, such that valid starting positions exist for each $\widetilde{\mathcal{P}}_u = \{\widetilde{P}_i \mid i \in I_u\}$.
LinFFP($\widetilde{\mathcal{P}}$)**:**
Given a set of linear patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ where $\widetilde{P}_i = (B, n_i, d_i)$, does there exist a set of starting positions S such that $\widetilde{\mathcal{P}}(S)$ is a feasible template?
LinMaxTemp($\widetilde{\mathcal{P}}$)**:**
Given a set of linear patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ where $\widetilde{P}_i = (B, n_i, d_i)$, determine a largest subset $I_1 \subseteq I$ such that valid starting positions exist for $\widetilde{\mathcal{P}}_1 = \{\widetilde{P}_i \mid i \in I_1\}$.

Comment 3.1.3: Consider the Periodic Scheduling Problem (PSP) of Korst, et al. [17]. In this problem we are given a set T of n periodic tasks, where each task $i \in T$ has a period $p(i) \in \mathbb{N}$ and an execution time $e(i) \in \mathbb{N}$ with $e(i) \leq p(i)$. We are also given a set of identical processors, and a positive integer k. Tasks are assigned to processors and a starting time $s(i)$ is given for each task i with the requirement that no processor is expected to run two tasks at the same time. The problem is to find such an assignment that uses at most k processors. Note that the periodic tasks from the PSP correspond to the linear patterns of Definition 3.1.1 with $p(i) = \dfrac{B}{n_i}$[1] and $e(i) = d_i$.[2]. Moreover, the processors of the PSP are analogous to templates in LinMinTemp and a task's starting time $s(i)$ corresponds to a pattern's starting position s_i. Also note that

$$\begin{aligned}\big(p(i), p(j)\big) &= \left(\frac{B}{n_i}, \frac{B}{n_j}\right) \\ &= \frac{B}{[n_i, n_j]}.\end{aligned}$$

Thus, LinMinTemp is equivalent to PSP and this allows us to state the following theorem.

Theorem 3.1.4 [3]**(Korst, Aarts, Lenstra & Wessels [18])**
Given a set of linear patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ *where* $\widetilde{P}_i = (B, n_i, d_i)$*, then* $\{\widetilde{P}_i(\overline{s}_i) \mid i \in I\}$ *is a feasible template iff*

$$d_i \leq (\overline{s}_j - \overline{s}_i) \bmod \frac{B}{[n_i, n_j]} \leq \frac{B}{[n_i, n_j]} - d_j, \quad \forall i \neq j. \tag{3.1.1}$$

As with generalized patterns, it is useful to define the recurring value

$$B_{ij} = \frac{B}{[n_i, n_j]}.$$

[1] While $\dfrac{B}{n_i}$ may not satisfy the requirement that $p(i)$ be integer, one can scale all patterns by $\dfrac{[n_1, n_2, ..., n_m]}{B}$ to correct this.

[2] Again, the d_i's may not satisfy the integrality constraint imposed by Korst, et al. on $e(i)$, but this can be corrected by insisting that the d_i's be rational (and then scaling them appropriately to achieve integrality). However, even with positive and real-valued $e(i)$'s all proofs in [17] would remain valid.

[3] *cf.* Corollary 2.2.12

Note that Theorem 3.1.4 is equivalent to

Theorem 3.1.5 [4] *Suppose $\widetilde{\mathcal{P}} = \{\widetilde{P_i} \mid i \in I\}$ is a set of linear patterns where $\widetilde{P_i} = (B, n_i, d_i)$, and let $\preceq$ be a total order on I. Then $\{\widetilde{P_i}(\overline{s}_i) \mid i \in I\}$ is a feasible template iff there exist $k_{ij} \in \mathbb{Z}$, $i \prec j \in I$, such that*

$$k_{ij}B_{ij} + d_i \leq \overline{s}_j - \overline{s}_i \leq (k_{ij}+1)B_{ij} - d_j, \quad \forall i \prec j. \tag{3.1.2}$$

The similarities between (3.1.2) and (2.3.1) lead to the following:

Comment 3.1.6: Recall that we are dealing with patterns on the same outer circumference. Consider inequality (2.2.7) from Corollary 2.2.16:

$$k\frac{2\pi}{[n_i,n_j]} + 2\arcsin\Big(\frac{\rho_i+\rho_j}{2R}\Big) \leq \varphi_j - \varphi_i \leq (k+1)\frac{2\pi}{[n_i,n_j]} - 2\arcsin\Big(\frac{\rho_i+\rho_j}{2R}\Big).$$

Now, using the identity function to approximate the arcsine, we get

$$\begin{aligned}
k\frac{2\pi}{[n_i,n_j]} + 2\frac{\rho_i+\rho_j}{2R} &\leq \varphi_j - \varphi_i \leq (k+1)\frac{2\pi}{[n_i,n_j]} - 2\frac{\rho_i+\rho_j}{2R} \\
k\frac{2\pi R}{[n_i,n_j]} + \rho_i + \rho_j &\leq R\varphi_j - R\varphi_i \leq (k+1)\frac{2\pi R}{[n_i,n_j]} - \rho_i - \rho_j \\
k\frac{2\pi R}{[n_i,n_j]} + 2\rho_i &\leq (R\varphi_j - \rho_j) - (R\varphi_i - \rho_i) \leq (k+1)\frac{2\pi R}{[n_i,n_j]} - 2\rho_j.
\end{aligned}$$

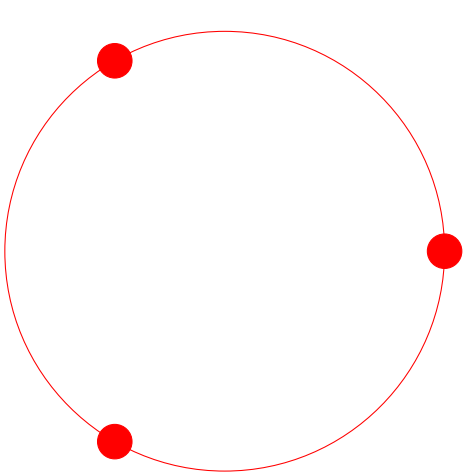
Note that $2\pi R$ is the circumference of the outer circle B. Additionally, $2\rho_i$ is the diameter of the disks of P_i, but we can approximate it using the arc length d_i of the outer circumference contained in each disk. Finally, recall that $R\varphi_i$ is the length of the arc on the outer circumference between the arbitrary zero point and the center of the first disk, which translates to $s_i + \dfrac{d_i}{2}$. This gives us

$$\begin{aligned}
k\frac{B}{[n_i,n_j]} + d_i \leq \Big(s_j + \frac{d_j}{2} - \frac{d_j}{2}\Big) - \Big(s_i + \frac{d_i}{2} - \frac{d_i}{2}\Big) &\leq (k+1)\frac{B}{[n_i,n_j]} - d_j \\
k\frac{B}{[n_i,n_j]} + d_i \leq s_j - s_i &\leq (k+1)\frac{B}{[n_i,n_j]} - d_j.
\end{aligned}$$

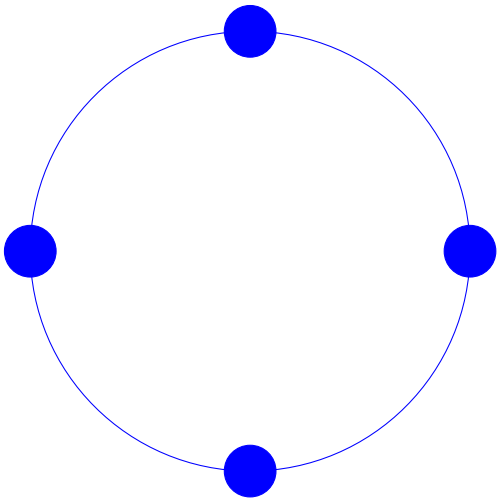
Thus the problem of fitting linear patterns (which are equivalent to periodic tasks) approximates the general problem of fitting patterns of disks on the same outer circumference.

[4] *cf.* Theorem 2.3.1

Example 3.1.7: Consider the set $\mathcal{P}^3$ containing patterns $P_1 = \left(\frac{20}{\pi}, 3, \frac{20}{\pi}\sin\frac{\pi}{40}\right) \approx (6.3662, 3, 0.4995)$ and $P_2 = \left(\frac{20}{\pi}, 4, \frac{20}{\pi}\sin\frac{3\pi}{80}\right) \approx (6.3662, 4, 0.7483)$ shown in Figures 3.1.1(a) and 3.1.1(b) respectively.



3.1.1(a): Generalized pattern P_1



3.1.1(b): Generalized pattern P_2

Figure 3.1.1: $\mathcal{P}^3$: Generalized patterns

We will approximate these patterns with the linear patterns $\widetilde{P_1} = (40, 3, 1)$ and $\widetilde{P_1} = (40, 4, 1.5)$, shown in Figures 3.1.2(a) and 3.1.2(b), which will be contained in set $\widetilde{\mathcal{P}^3}$.

3.1.2(a): Linear pattern $\widetilde{P_1}$

3.1.2(b): Linear pattern $\widetilde{P_2}$

Figure 3.1.2: $\widetilde{\mathcal{P}^3}$: Linear patterns

Using Theorem 3.1.4, we verify that the starting positions $s_1 = 0$, $s_2 = 1.3$ are valid for $\widetilde{P}_1, \widetilde{P}_2$. First we calculate the value

$$B_{12} = \frac{B}{[n_1, n_2]} = \frac{40}{[3,4]} = \frac{40}{12} \approx 3.333.$$

Next

$$(s_2 - s_1) \bmod B_{12} = (1.3 - 0) \bmod 3.333 = 1.3.$$

Thus

$$1 \leq 1.3 \leq 1.833$$

$$1 \leq 1.3 \leq 3.333 - 1.5$$

$$d_1 \leq (s_2 - s_1) \bmod B_{12} \leq B_{12} - d_2$$

So the template $\{\widetilde{P}_1(0), \widetilde{P}_2(1.3)\}$ shown in Figure 3.1.3 is feasible.

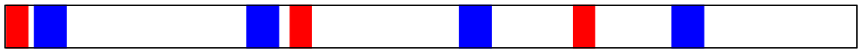


Figure 3.1.3: $\widetilde{\mathcal{P}}^3$: The feasible template $\{\widetilde{P}_1(0), \widetilde{P}_2(1.3)\}$

Because of the similarity between Theorem 2.3.1 and Theorem 3.1.5, we can derive the same results for linear patterns that we did for generalized patterns in Chapter 2. The proofs of these corollaries and theorems are identical to those given in Chapter 2, but with B_{ij} replacing β_{ij} and d_i and d_j replacing δ_{ij}. To illustrate this point, we give the corollary used to calculate the value of k_{ij} for inequality (3.1.2). Compare its proof to that of Corollary 2.2.9.

Corollary 3.1.8 [5] *Let linear patterns $\widetilde{P}_i = (B, n_i, d_i)$ and $\widetilde{P}_j = (B, n_j, d_j)$ be given. If $\overline{s}_i, \overline{s}_j$ are valid for $\{\widetilde{P}_i, \widetilde{P}_j\}$ then*

$$k_{ij} = \left\lfloor \frac{\overline{s}_j - \overline{s}_i - d_i}{B_{ij}} \right\rfloor$$

is the unique value that satisfies inequality (3.1.2) *from Theorem 3.1.5.*

[5] *cf.* Corollary 2.2.9

Proof: Assume that $\widetilde{P}_i(\overline{s}_i) \cap \widetilde{P}_j(\overline{s}_j) = \emptyset$. Thus, by Theorem 3.1.5, there exists $k_{ij} \in \mathbb{Z}$ such that

$$k_{ij}B_{ij} + d_i \leq \overline{s}_j - \overline{s}_i \leq (k_{ij}+1)B_{ij} - d_j$$

$$d_i \leq \overline{s}_j - \overline{s}_i - k_{ij}B_{ij} \leq B_{ij} - d_j$$

$$d_i - (\overline{s}_j - \overline{s}_i) \leq -k_{ij}B_{ij} \leq B_{ij} - d_j - (\overline{s}_j - \overline{s}_i)$$

$$\overline{s}_j - \overline{s}_i + d_j - B_{ij} \leq k_{ij}B_{ij} \leq \overline{s}_j - \overline{s}_i - d_i$$

$$\frac{\overline{s}_j - \overline{s}_i + d_j}{B_{ij}} - 1 \leq k_{ij} \leq \frac{\overline{s}_j - \overline{s}_i - d_i}{B_{ij}}.$$

As a result, we must have at least one integer k_{ij} inside the interval

$$\left\lfloor \frac{\overline{s}_j - \overline{s}_i + d_j}{B_{ij}} - 1,\ \frac{\overline{s}_j - \overline{s}_i - d_i}{Bij} \right\rfloor. \tag{3.1.3}$$

However, note that since $d_i, d_j > 0$,

$$\frac{\overline{s}_j - \overline{s}_i - d_i}{B_{ij}} < \frac{\overline{s}_j - \overline{s}_i + d_j}{B_{ij}}.$$

From this it follows that interval (3.1.3) has length strictly less than 1 and thus contains only one integer whose value is

$$k_{ij} = \left\lceil \frac{\overline{s}_j - \overline{s}_i + d_j}{B_{ij}} - 1 \right\rceil = \left\lfloor \frac{\overline{s}_j - \overline{s}_i - d_i}{B_{ij}} \right\rfloor. \tag{3.1.4}$$

■

As the proof above shows, proving statements about linear patterns simply involves rewriting the equivalent proof for generalized patterns and replacing the appropriate parameters. As a result of this, we state the following corollaries without proof. These provide a series of tests that can be used to determine whether two patterns overlap. The first result calculates the value of k_{ij} for a pair of starting positions and tests whether or not they are valid.

Corollary 3.1.9 [6] *Let linear patterns $\widetilde{P}_i = (B, n_i, d_i)$ and $\widetilde{P}_j = (B, n_j, d_j)$ be given. Then $\overline{s}_i, \overline{s}_j$ are valid for $\{\widetilde{P}_i, \widetilde{P}_j\}$ iff*

$$\frac{\overline{s}_j - \overline{s}_i + d_j}{B_{ij}} - 1 \leq \left\lfloor \frac{\overline{s}_j - \overline{s}_i - d_i}{B_{ij}} \right\rfloor.$$

[6] *cf.* Corollary 2.2.11

Next, we can test the validity of starting positions using the modulo function.

Corollary 3.1.10 [7] *Let linear patterns $\widetilde{P}_i = (B, n_i, d_i)$ and $\widetilde{P}_j = (B, n_j, d_j)$ be given. Then $\overline{s}_i, \overline{s}_j$ are valid for $\{\widetilde{P}_i, \widetilde{P}_j\}$ iff*

$$(\overline{s}_j - \overline{s}_i - d_i) \bmod B_{ij} \leq B_{ij} - d_i - d_j. \tag{3.1.5}$$

Next, given two patterns we can test whether or not valid starting positions even exist.

Corollary 3.1.11 [8]**(Korst, Aarts, Lenstra & Wessels [18])** *Let linear patterns $\widetilde{P}_i = (B, n_i, d_i)$ and $\widetilde{P}_j = (B, n_j, d_j)$ be given. Then $\mathcal{S}\big(\{\widetilde{P}_i, \widetilde{P}_j\}\big) \neq \emptyset$ iff*

$$B_{ij} \geq d_i + d_j.$$

We can also bound the s_i and k_{ij} values in the same fashion as before.

Theorem 3.1.12 [9] *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be given with $\widetilde{P}_i = (B, n_i, d_i)$. Let $\preceq$ be a total order on the set I. If $\mathcal{S}(\widetilde{\mathcal{P}}) \neq \emptyset$ then there exist*

$$k_{ij} \in \mathbb{Z} \cap \left[-\frac{n_j}{(n_i, n_j)},\ \frac{n_i}{(n_i, n_j)} - 1 \right], \quad \forall i \prec j$$

and $\hat{s}_i \in \left[0, \dfrac{B}{n_i}\right)$ for all $i \in I$ such that

$$k_{ij} B_{ij} + d_i \leq \hat{s}_j - \hat{s}_i \leq (k_{ij} + 1) B_{ij} - d_j, \quad \forall i \prec j.$$

Comment 3.1.13: Using the same transformation applied to the corollaries above, we can modify the mixed integer programs (MIPs) developed in Section 2.3.2 so that they apply to linearized patterns. In fact, because of the similarities between Theorem 2.3.1 and Theorem 3.1.5, this can be achieved with relatively little effort. All references to φ_i and φ_j need to be changed to s_i and s_j, their bounds amended to be compatible with Theorem 3.1.12, and all references to β_{ij} and δ_{ij} should

[7] *cf.* Corollary 2.2.13
[8] *cf.* Corollary 2.2.14
[9] *cf.* Theorem 2.3.4

be replaced with B_{ij}, d_i and d_j appropriately. This results in the following three formulations:

$$
\begin{aligned}
\textsc{Fit}(\widetilde{\mathcal{P}}): \qquad & k_{ij}B_{ij} + s_i - s_j \leq -d_i && \forall i \prec j \\
& s_j - s_i - k_{ij}B_{ij} \leq B_{ij} - d_j && \forall i \prec j \\
& 0 \leq s_i \leq \frac{B}{n_i} && \forall i \in I \\
& -\frac{n_j}{(n_i, n_j)} \leq k_{ij} \leq \frac{n_i}{(n_i, n_j)} - 1 && \forall i \prec j \\
& k_{ij} \in \mathbb{Z} && \forall i \prec j \\
& \varphi_i \in \mathbb{R} && \forall i \in I,
\end{aligned}
$$

$$
\begin{aligned}
\textsc{MinTemplates}(\mathcal{P}, \preceq, U): \min & \sum_{u \in U} z_u \\
\text{s.t. } & k_{ij}B_{ij} + s_i - s_j + x_{iu}d_i + x_{ju}d_i \leq d_i && \forall i \prec j, u \in U \\
& s_j - s_i - k_{ij}B_{ij} + x_{iu}d_j + x_{ju}d_j \leq B_{ij} + d_j && \forall i \prec j, u \in U \\
& \sum_{u \in U} x_{iu} = 1 && \forall i \in I \\
& x_{iu} - z_u \leq 0 && \forall i \in I, u \in U \\
& 0 \leq s_i \leq \frac{B}{n_i} && \forall i \in I \\
& -\frac{n_j}{(n_i, n_j)} \leq k_{ij} \leq \frac{n_i}{(n_i, n_j)} - 1 && \forall i \prec j \\
& k_{ij} \in \mathbb{Z} && \forall i \prec j \\
& \varphi_i \in \mathbb{R} && \forall i \in I \\
& x_{iu} \in \{0, 1\} && \forall i \in I, u \in U \\
& z_u \geq 0 && \forall u \in U,
\end{aligned}
$$

and

$$\begin{aligned}
&\textsc{MaxTemplate}(\widetilde{\mathcal{P}}, \preceq) : \max \sum_{i \in I} x_i \\
&\text{s.t. } k_{ij}B_{ij} + s_i - s_j + x_i d_i + x_j d_i \leq d_i && \forall i \prec j \\
&\qquad s_j - s_i - k_{ij}B_{ij} + x_i d_j + x_j d_j \leq B_{ij} + d_j && \forall i \prec j \\
&\qquad 0 \leq s_i \leq \frac{B}{n_i} && \forall i \in I \\
&\qquad -\frac{n_j}{(n_i, n_j)} \leq k_{ij} \leq \frac{n_i}{(n_i, n_j)} - 1 && \forall i \prec j \\
&\qquad k_{ij} \in \mathbb{Z} && \forall i \prec j \\
&\qquad \varphi_i \in \mathbb{R} && \forall i \in I \\
&\qquad x_i \in \{0, 1\} && \forall i \in I.
\end{aligned}$$

In the same spirit as the corollaries above, we can also apply the network formulation from Section 2.4 to linearized patterns.

Definition 3.1.14 [10] *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns where $\widetilde{P}_i = (B, n_i, d_i)$, let $\preceq$ be a total order on I, and let $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$. We define the network $G(\widetilde{\mathcal{P}}, \preceq, K) = (N, A, w)$ where*

$$\begin{aligned}
N &= I \\
A &= \big\{(i, j) \;\big|\; i \neq j \in I\big\} \\
w_{ij} &= \begin{cases} (k_{ij} + 1)B_{ij} - d_j, & i \prec j \\ -k_{ji}B_{ji} - d_j, & j \prec i. \end{cases}
\end{aligned}$$

Again, since the weights of $G(\widetilde{\mathcal{P}}, \preceq, K)$ are so similar to the weights of $G(\mathcal{P}, \preceq, K)$, the proofs of the next four theorems can be obtained from their corresponding proofs in Section 2.4 just by replacing the appropriate parameters. As a result, these will be stated without proof. The first theorem shows that valid starting positions can be obtained from the shortest path distances from (or to) a given root node in $G(\widetilde{\mathcal{P}}, \preceq, K)$.

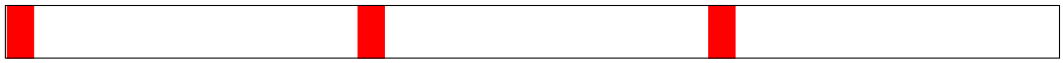
[10] *cf.* Definition 2.4.1

Theorem 3.1.15 [11] *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns, let $\preceq$ be a total order on I, and let $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$. If $\{\overline{s}_i \mid i \in I\}$ are shortest path distances from a specified root node in network $G(\widetilde{\mathcal{P}}, \preceq, K)$, then $\{\overline{s}_i \mid i \in I\} \in \mathcal{S}(\widetilde{\mathcal{P}})$. Also, if $\{\overline{t}_i \mid i \in I\}$ are valid shortest path distances to a specified root node in network $G(\widetilde{\mathcal{P}}, \preceq, K)$, then $\{-\overline{t}_i \mid i \in I\} \in \mathcal{S}(\widetilde{\mathcal{P}})$.*

The next theorem shows the connection between the existence of valid starting positions and negative (weight) cycles in $G(\widetilde{\mathcal{P}}, \preceq, K)$.

Theorem 3.1.16 [12] *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns and let $\preceq$ be a total order on I. $\mathcal{S}(\widetilde{\mathcal{P}}) \neq \emptyset$ iff there exists $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$ such that $G(\widetilde{\mathcal{P}}, \preceq, K)$ contains no negative (weight) cycle.*

Example 3.1.17: Consider the set $\widetilde{\mathcal{P}}^4$ containing linear patterns $\widetilde{P}_1 = (120, 3, 3)$, $\widetilde{P}_2 = (120, 4, 3)$, and $\widetilde{P}_3 = (120, 5, 3)$ shown in Figure 3.1.4.



3.1.4(a): Pattern $\widetilde{P}_1$

3.1.4(b): Pattern $\widetilde{P}_2$

3.1.4(c): Pattern $\widetilde{P}_3$

Figure 3.1.4: $\widetilde{\mathcal{P}}^4$: Fitting three linear patterns on a template

First we calculate the following values:

$$B_{12} = \frac{120}{[3,4]} = 10$$
$$B_{13} = \frac{120}{[3,5]} = 8$$
$$B_{23} = \frac{120}{[4,5]} = 6.$$

[11] *cf.* Theorem 2.4.2 and Corollary 2.4.3
[12] *cf.* Theorem 2.4.5

Since $d_1 = d_2 = d_3 = 3$ and $B_{ij} \geq 6 \geq d_i + d_j$ for all $i \neq j \in \{1, 2, 3\}$ Corollary 3.1.11 implies that valid starting positions exist for each pair of patterns. If we use the ordering $1 \prec 2 \prec 3$ and treat K as unknown, then the network $G(\widetilde{\mathcal{P}}, \preceq, K)$ is as shown in Figure 3.1.5.

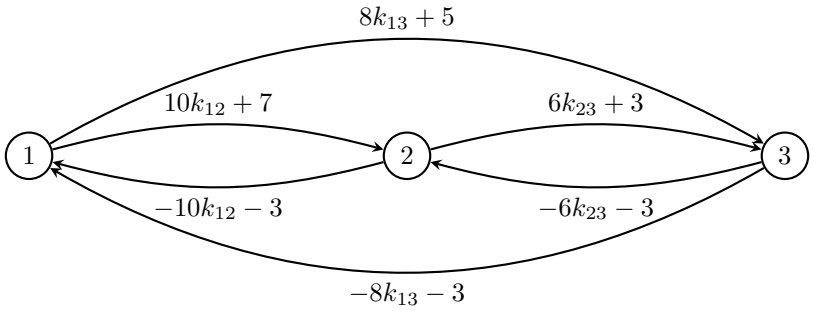


Figure 3.1.5: $\widetilde{\mathcal{P}}^4$: $G(\widetilde{\mathcal{P}}, \preceq, K)$

If we propose $\overline{K} = \{\overline{k}_{12} = 0, \overline{k}_{13} = 1, \overline{k}_{23} = 1\}$ then $G(\widetilde{\mathcal{P}}, \preceq, \overline{K})$ is shown in Figure 3.1.6.

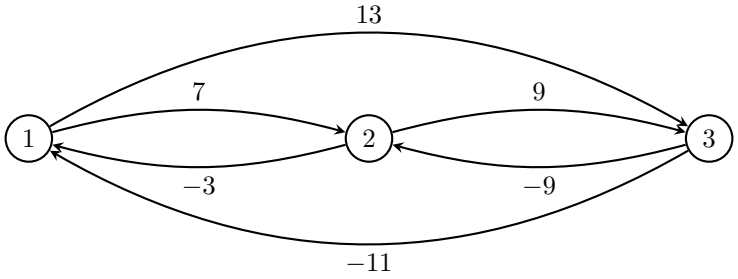


Figure 3.1.6: $\widetilde{\mathcal{P}}^4$: $G(\widetilde{\mathcal{P}}, \preceq, \overline{K})$

Finding shortest paths from node 1 gives (0, 4, 13) = $(\overline{s}_1, \overline{s}_2, \overline{s}_3)$. The corresponding solution is shown in Figure 3.1.7(a). Alternatively, finding shortest paths to node 1 gives (0, −3, −12) = $(-\overline{s}_1, -\overline{s}_2, -\overline{s}_3)$, with the corresponding solution is shown in Figure 3.1.7(b).

3.1.7(a): The feasible template $\{\widetilde{P}_1(0), \widetilde{P}_2(4), \widetilde{P}_3(13)\}$

3.1.7(b): The feasible template $\{\widetilde{P}_1(0), \widetilde{P}_2(3), \widetilde{P}_3(12)\}$

Figure 3.1.7: $\widetilde{\mathcal{P}}^4$: Solutions gained from the network

The next two theorems deal with the ordering $\preceq$ and its effect on $G(\widetilde{\mathcal{P}}, \preceq, K)$. The first shows that, given a change in the ordering $\preceq$ there is a corresponding change in K that will keep the network invariant.

Theorem 3.1.18 [13] *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns, let $\preceq$ be a total order on I, and let $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$. Let $p \prec q$ be such that there does not exist $r \in I$ satisfying $p \prec r \prec q$ (i.e., p and q are adjacent w.r.t. $\preceq$). Define $\preceq'$ by $q \prec' p$ and*

$$i \prec' j \Leftrightarrow i \prec j, \quad \forall \{i, j\} \neq \{p, q\} \subseteq I$$

i.e., $\preceq'$ is exactly the same as $\preceq$ except that the order of p and q is switched. Also, let $K' = \{k'_{ij} \mid i \prec' j\}$ where

$$k'_{ij} = \begin{cases} -k_{ji} - 1, & i = q,\ j = p \\ k_{ij}, & \text{else.} \end{cases}$$

Then $G(\widetilde{\mathcal{P}}, \preceq, K) = G(\widetilde{\mathcal{P}}, \preceq', K')$.

As with Theorem 2.4.6, Theorem 3.1.18 is sufficient to show that, given network $G(\widetilde{\mathcal{P}}, \preceq, K)$ and any other total order $\preceq'$ on I, there exists K' such that

$$G(\widetilde{\mathcal{P}}, \preceq, K) = G(\mathcal{P}, \preceq', K').$$

[13] *cf.* Theorem 2.4.6

Finally, the following theorem details the special case of inverse orderings.

Theorem 3.1.19 [14] *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns. Let $\preceq_1$ and $\preceq_2$ be total orderings on I such that*

$$i \preceq_1 j \Leftrightarrow j \preceq_2 i, \quad \forall i, j \in I,$$

i.e., $\preceq_1$ and $\preceq_2$ are inverses of each other. Let $K^1 = \{k^1_{ij} \in \mathbb{Z} \mid i \prec_1 j\}$ be given and let $K^2 = \{k^2_{ij} \in \mathbb{Z} \mid i \prec_2 j\}$ be defined by $k^2_{ij} = k^1_{ji}$ for all $i \prec_2 j$. Then $G(\widetilde{\mathcal{P}}, \preceq_1, K^1)$ has no negative weight cycle iff $G(\widetilde{\mathcal{P}}, \preceq_2, K^2)$ has no negative weight cycle.

3.2 Special Orderings

Let us consider the network $G(\widetilde{\mathcal{P}}, \preceq, K)$. We know that given an appropriate set K, we can find shortest path distances in the network and use them as valid starting positions. However, the question remains as to how to find such a set K. In this section we investigate conditions that enable the use of a very specific set K which guarantees that shortest path distances exist in $G(\widetilde{\mathcal{P}}, \preceq, K)$.

In order for $G(\widetilde{\mathcal{P}}, \preceq, K)$ to have shortest path distances, it must contain no negative cycles. Let $\mathcal{C}$ be the set of all directed cycles in $G(\widetilde{\mathcal{P}}, \preceq, K)$. Consider $C \in \mathcal{C}$. If we define, as in the proof for Theorem 2.4.5

$$C_{\prec} := \{(i, j) \in C \mid i \prec j\}$$
$$C_{\succ} := \{(i, j) \in C \mid j \prec i\}$$

[14] *cf.* Theorem 2.4.7

then the total weight of C is

$$\begin{aligned} w[C] &= \sum_{(i,j)\in C} w_{ij} \\ &= \sum_{(i,j)\in C_\prec} [(k_{ij}+1)B_{ij} - d_j] + \sum_{(i,j)\in C_\succ} [-k_{ji}B_{ji} - d_j] \\ &= \sum_{(i,j)\in C_\prec} (k_{ij}+1)B_{ij} - \sum_{(i,j)\in C_\prec} d_j - \sum_{(i,j)\in C_\succ} k_{ji}B_{ji} - \sum_{(i,j)\in C_\succ} d_j \\ &= \sum_{(i,j)\in C_\prec} (k_{ij}+1)B_{ij} - \sum_{(i,j)\in C_\succ} k_{ji}B_{ji} - \sum_{j\in C} d_j. \end{aligned} \tag{3.2.1}$$

Thus, for the network to have no negative cycles, we need

$$\begin{aligned} w[C] &\geq 0 \\ \sum_{(i,j)\in C_\prec} (k_{ij}+1)B_{ij} - \sum_{(i,j)\in C_\succ} k_{ji}B_{ji} - \sum_{j\in C} d_j &\geq 0 \\ \sum_{(i,j)\in C_\prec} k_{ij}B_{ij} + \sum_{(i,j)\in C_\prec} B_{ij} - \sum_{(i,j)\in C_\succ} k_{ji}B_{ji} - \sum_{j\in C} d_j &\geq 0 \\ \sum_{(i,j)\in C_\prec} k_{ij}B_{ij} - \sum_{(i,j)\in C_\succ} k_{ji}B_{ji} &\geq \sum_{j\in C} d_j - \sum_{(i,j)\in C_\prec} B_{ij}. \end{aligned} \tag{3.2.2}$$

Finding k_{ij}'s that satisfy this inequality for all cycles in $G(\widetilde{\mathcal{P}}, \preceq, K)$ is not easy. Instead we consider a very specific set K and find simple conditions such that there are no negative cycles in $G(\widetilde{\mathcal{P}}, \preceq, K)$. Namely, let $K_0 = \{k_{ij} = 0 \mid i \prec j\}$. This makes the left-hand side of the inequality (3.2.2) zero, so $G(\widetilde{\mathcal{P}}, \preceq, K_0)$ has no negative cycles iff

$$\begin{aligned} 0 &\geq \sum_{j\in C} d_j - \sum_{(i,j)\in C_\prec} B_{ij} \\ \sum_{(i,j)\in C_\prec} B_{ij} &\geq \sum_{j\in C} d_j, \qquad \forall C \in \mathcal{C}. \end{aligned} \tag{3.2.3}$$

Theorem 3.1.18 implies that the existence of valid starting positions is not dependent on the ordering. If the ordering is changed, then we just have to change the set K appropriately to maintain the network structure. However, we now wish to keep the set K_0 fixed. Consequently, a change in $\preceq$ will change the network structure and thus the weights of the cycles. Indeed, examples can be constructed to show that $G(\widetilde{\mathcal{P}}, \preceq, K_0)$ having no negative cycles does not guarantee that $G(\widetilde{\mathcal{P}}, \preceq', K_0)$

has no negative cycles for every $\preceq'$. Thus, $G(\widetilde{\mathcal{P}}, \preceq, K_0)$ having no negative cycles is dependent on the given total ordering $\preceq$, which is consistent with the fact that inequalities (3.2.3) are also dependent on $\preceq$.

Given a total ordering $\preceq$ on I, we can check the inequalities (3.2.3) to determine whether $G(\widetilde{\mathcal{P}}, \preceq, K_0)$ has no negative cycles. However, there is one inequality per cycle in $G(\widetilde{\mathcal{P}}, \preceq, K_0)$ and there are $O\big(|I|!\big)$ cycles in the network. We can in fact reduce the effort to only checking $O\big(|I|^2\big)$ inequalities, but first we need the following lemma.

Lemma 3.2.1 *Let $G = (N, A)$ be a directed graph and let $\preceq$ be a total order on the elements of N. Given any directed cycle C in G and any node m in cycle C, there exists an arc $(p, q) \in C$ such that*

$$p \preceq m \preceq q.$$

Proof: Let us assume that the nodes in cycle C are visited in the order

$$i_0, i_1, ..., i_\ell$$

and let $m = i_k$ be any node in C. Now express the cycle C as the list of nodes visited in the following order:

$$\big[i_{k+1}, ..., i_\ell, i_0, ..., i_{k-1}, m\big].$$

Let $q = i_w$ be the first element in this list such that $m \preceq q$ and let $p = i_{w-1}$. Then

$$p \preceq m \preceq q$$

and $(p, q) \in C$. ■

We now proceed to reduce the number of required inequalities in (3.2.3).

Theorem 3.2.2 *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns and let $\preceq$ be a total order on I. The network $G(\widetilde{\mathcal{P}}, \preceq, K_0)$ has no negative cycles iff*

$$B_{ij} \geq \sum_{i \preceq \ell \preceq j} d_\ell, \quad \forall i \prec j.$$

Proof:

($\Leftarrow$)

Assume that

$$B_{ij} \geq \sum_{i \preceq \ell \preceq j} d_\ell, \quad \forall i \prec j.$$

Now consider any cycle C in G. Using equation (3.2.1) for the set K_0 gives

$$\begin{aligned}
w(C) &= \sum_{(i,j) \in C_\prec} (k_{ij}+1)B_{ij} - \sum_{(i,j) \in C_\succ} k_{ji}B_{ji} - \sum_{j \in C} d_j \\
&= \sum_{(i,j) \in C_\prec} B_{ij} - \sum_{j \in C} d_j \\
&= \sum_{(i,j) \in C_\prec} B_{ij} - \sum_{(i,j) \in C_\prec} \sum_{i \preceq \ell \preceq j} d_\ell + \sum_{(i,j) \in C_\prec} \sum_{i \preceq \ell \preceq j} d_\ell - \sum_{j \in C} d_j \\
&= \sum_{(i,j) \in C_\prec} \left[B_{ij} - \sum_{i \preceq \ell \preceq j} d_\ell \right] + \sum_{(i,j) \in C_\prec} \sum_{i \preceq \ell \preceq j} d_\ell - \sum_{j \in C} d_j.
\end{aligned}$$

From the assumption

$$B_{ij} \geq \sum_{i \preceq \ell \preceq j} d_\ell, \quad \forall i \prec j$$

the first summation is non-negative. Thus, if we can show that

$$\sum_{(i,j) \in C_\prec} \sum_{i \preceq \ell \preceq j} d_\ell - \sum_{j \in C} d_j \geq 0$$

then the cycle weight will be non-negative. Now, from Lemma 3.2.1, we know that for all $m \in C$, there must exist $(i,j) \in C_\prec$ such that $i \preceq m \preceq j$. Thus

$$\{m \mid m \in C\} \subseteq \bigcup_{(i,j) \in C_\prec} \{\ell \mid i \preceq \ell \preceq j\}$$

giving

$$\sum_{(i,j)\in C_{\prec}} \sum_{i \preceq \ell \preceq j} d_\ell \geq \sum_{j \in C} d_j.$$

Thus $w(C) \geq 0$.

($\Rightarrow$)

Assume that $G(\widetilde{\mathcal{P}}, \preceq, K_0)$ has no negative cycles. Let $\preceq$ be a total order on I and let $p \prec q$ be given. Let the indices of I between p and q be denoted as

$$p \prec \ell_1 \prec \ell_2 \prec \cdots \prec \ell_{m-1} \prec \ell_m \prec q.$$

Let C be the cycle in G that visits the nodes

$$[q, \ell_m, \ell_{m-1}, ..., \ell_2, \ell_1, p, q].$$

The total weight of C relative to K_0 is from equation (3.2.1)

$$\begin{aligned} w(C) &= \sum_{(i,j)\in C_{\prec}} (k_{ij}+1)B_{ij} - \sum_{(i,j)\in C_{\succ}} k_{ji}B_{ji} - \sum_{j\in C} d_j \\ &= \sum_{(i,j)\in C_{\prec}} B_{ij} - \sum_{j\in C} d_j \\ &= B_{pq} - \sum_{p \preceq \ell \preceq q} d_\ell. \end{aligned}$$

Since $w[C]$ is non-negative by assumption, we have

$$B_{pq} \geq \sum_{p \preceq \ell \preceq q} d_\ell.$$

■

This leads to the following definition.

Definition 3.2.3 *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns with $\widetilde{P}_i = (B, n_i, d_i)$ for all $i \in I$. A total order $\preceq$ on I is called zero-feasible for $\widetilde{\mathcal{P}}$ if*

$$B_{ij} \geq \sum_{i \preceq \ell \preceq j} d_\ell, \quad \forall i \prec j \tag{3.2.4}$$

and the pair $(I, \preceq)$ will be called a zero-feasible chain.

Consider $i \prec j$ and recall from Comment 2.2.5 that $k_{ij} = 0$ implies that the first interval of $\widetilde{P}_j$ lies between the first and second intervals of the pattern $\widetilde{P}_{ij} = (B, [n_i, n_j], d_i)$. Thus the first interval of $\widetilde{P}_j$ must closely follow the first interval of $\widetilde{P}_i$. Since this must hold for all $i \prec j$ we can state the following.

Theorem 3.2.4 *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns with $\widetilde{P}_i = (B, n_i, d_i)$ for all $i \in I$ and let $\preceq$ be a total order on I. If $\preceq$ is zero-feasible then the set of starting positions*

$$\left\{ s_i = \sum_{\ell \prec i} d_\ell \,\middle|\, i \in I \right\} \tag{3.2.5}$$

is in $\widetilde{\mathcal{P}}(\mathcal{S})$.

Note that, if i is the minimum element of I with respect to $\preceq$, then the summation $\sum_{\ell \prec i} d_\ell$ is null. We will assume throughout this dissertation that a null summation has value zero.

Proof: Assume that $\preceq$ is zero-feasible, so

$$B_{ij} \geq \sum_{i \preceq \ell \preceq j} d_\ell, \quad \forall i \prec j.$$

Recall from Theorem 3.1.5 that $\{\overline{s}_i \mid i \in I\}$ is in $\mathcal{S}(\widetilde{\mathcal{P}})$ if there exist $k_{ij} \in \mathbb{Z}$ such that

$$k_{ij} B_{ij} + d_i \leq \overline{s}_j - \overline{s}_i \leq (k_{ij} + 1) B_{ij} - d_j, \quad \forall i \prec j.$$

Let $k_{ij} = 0$ for all $i \prec j$ and note that

$$\sum_{i \preceq \ell \preceq j} d_\ell \leq B_{ij}$$
$$\Leftrightarrow \sum_{i \prec \ell \prec j} d_\ell \leq B_{ij} - d_i - d_j.$$

Now

$$0 \leq \sum_{i \prec \ell \prec j} d_\ell \leq B_{ij} - d_i - d_j$$
$$0 \leq \sum_{\ell \prec j} d_\ell - \sum_{\ell \preceq i} d_\ell \leq B_{ij} - d_i - d_j$$
$$0 \leq \sum_{\ell \prec j} d_\ell - \sum_{\ell \prec i} d_\ell - d_i \leq B_{ij} - d_i - d_j$$
$$d_i \leq \sum_{\ell \prec j} d_\ell - \sum_{\ell \prec i} d_\ell \leq B_{ij} - d_j$$
$$0 \cdot B_{ij} + d_i \leq \overline{s}_j - \overline{s}_i \leq (0+1)B_{ij} - d_j$$
$$k_{ij}B_{ij} + d_i \leq \overline{s}_j - \overline{s}_i \leq (k_{ij}+1)B_{ij} - d_j.$$

This implies that the $\overline{s}_i$'s defined by (3.2.5) are valid starting positions for $\widetilde{\mathcal{P}}$. ■

Alternatively, we can prove Theorem 3.2.4 as follows.

Proof: Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ and $\preceq$ be given. Assume that $\preceq$ is zero-feasible for $\widetilde{\mathcal{P}}$. Thus, by Theorem 3.2.2, $G(\widetilde{\mathcal{P}}, \preceq, K_0)$ has no negative cycles. Let the elements of I be listed according to $\preceq$ as

$$i_0 \prec i_1 \prec i_2 \prec \cdots \prec i_m$$

and define the path Q in $G(\widetilde{\mathcal{P}}, \preceq, K_0)$ by

$$i_m \to i_{m-1} \to i_{m-2} \to \cdots \to i_1 \to i_0.$$

Note that in Q there is a path Q_r from any node i_r to i_0 which has total length

$$\begin{aligned} w[Q_r] &= w_{i_r i_{r-1}} + w_{i_{r-1} i_{r-2}} + \cdots + w_{i_1 i_0} \\ &= -d_{r-1} - d_{r-2} - \cdots - d_0 \\ &= -\sum_{\ell \prec r} d_\ell. \end{aligned}$$

We claim that Q defines a shortest path tree into node i_0 and thus the values $t_r = w[Q_r]$ are the shortest path distances from any node r to node i_0. First note that $t_{i_0} = 0$. Now, let $p \prec q \in I$ be given. Then

$$\begin{aligned} t_p - t_q &= -\sum_{\ell \prec p} d_\ell + \sum_{\ell \prec q} d_\ell \\ &= \sum_{p \preceq \ell \prec q} d_\ell \\ &= \sum_{p \preceq \ell \preceq q} d_\ell - d_q \\ &\leq B_{pq} - d_q \\ &= w_{pq} \end{aligned} \qquad \begin{aligned} t_q - t_p &= -\sum_{\ell \prec q} d_\ell + \sum_{\ell \prec p} d_\ell \\ &= -\sum_{p \preceq \ell \prec q} d_\ell \\ &= -d_p - \sum_{p \prec \ell \prec q} d_\ell \\ &\leq -d_p \\ &= w_{qp}. \end{aligned}$$

Thus, the t_r's satisfy the optimality conditions for shortest paths to a specified node and by Theorem 3.1.15

$$\left\{ s_i = -t_i = \sum_{\ell \prec i} d_\ell \;\middle|\; i \in I \right\} \in \mathcal{S}(\widetilde{\mathcal{P}}).$$

■

Thus zero-feasible orderings have the property that we can arrange the patterns in $\widetilde{\mathcal{P}}$ on a feasible template where the first intervals of all patterns form a contiguous block in the order given by $\preceq$. The periodic scheduling analog of this situation would involve performing each task in the order $\preceq$ a single time before performing any task a second time. To illustrate, we present the following example.

Example 3.2.5: Consider the set $\widetilde{\mathcal{P}}^4$ from Example 3.1.17 which contains linear patterns $\widetilde{P}_1 = (120, 3, 3)$, $\widetilde{P}_2 = (120, 4, 3)$, and $\widetilde{P}_3 = (120, 5, 3)$. We showed earlier that

$$B_{12} = 10 \geq 6 = d_1 + d_2$$

$$B_{23} = 6 \geq 6 = d_1 + d_2.$$

However, since

$$B_{13} = 8 < 9 = d_1 + d_2 + d_3,$$

the ordering $1 \prec 2 \prec 3$ is not zero-feasible. However, consider the order $1 \prec' 3 \prec' 2$. Then

$$B_{13} = 8 \geq 6 = d_1 + d_3$$

$$B_{32} = 6 \geq 6 = d_1 + d_2$$

$$B_{12} = 10 \geq 9 = d_1 + d_3 + d_2$$

and so $\prec'$ is zero-feasible. This implies that $G(\widetilde{\mathcal{P}}, \preceq', \overline{K}_0)$ has no negative cycles, and that $\{\widetilde{P}_1(0), \widetilde{P}_3(3), \widetilde{P}_2(6)\}$ shown in Figure 3.2.1, is feasible.

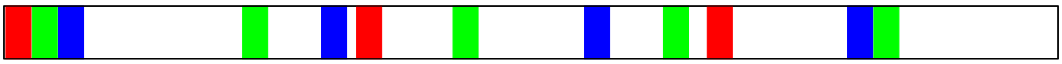


Figure 3.2.1: $\widetilde{\mathcal{P}}^4$: The feasible template $\{\widetilde{P}_1(0), \widetilde{P}_3(3), \widetilde{P}_2(6)\}$

The relationship detailed in Theorem 3.2.4 is not unique to zero-feasible orderings and this leads to the following definition.

Definition 3.2.6 *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns with $\widetilde{P}_i = (B, n_i, d_i)$ for all $i \in I$. A total order $\preceq$ on I is called a contiguous ordering for $\widetilde{\mathcal{P}}$ if*

$$\left\{ \overline{s}_i = \sum_{\ell \prec i} d_\ell \;\middle|\; i \in I \right\} \in \mathcal{S}(\widetilde{\mathcal{P}})$$

and the pair $(I, \preceq)$ will be called a contiguous chain.

Theorem 3.2.4 implies that any zero-feasible ordering is also a contiguous ordering. The reverse is not true, but the following theorem shows how the definition of zero-feasibility can be generalized to characterize contiguous orderings.

Theorem 3.2.7 *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be a set of linear patterns where $\widetilde{P}_i = (B, n_i, d_i)$ and let $\preceq$ be a total order on I. Then $\preceq$ is a contiguous ordering for $\widetilde{\mathcal{P}}$ iff*

$$\left(\sum_{i \prec \ell \prec j} d_\ell \right) \bmod B_{ij} \leq B_{ij} - d_i - d_j, \quad \forall i \prec j. \tag{3.2.6}$$

Proof: Corollary 3.1.10 guarantees that $\left\{ \overline{s}_i = \sum_{\ell \prec i} d_\ell \,\middle|\, i \in I \right\} \in \mathcal{S}(\widetilde{\mathcal{P}})$ iff

$$\begin{aligned}
(\overline{s}_j - \overline{s}_i - d_i) \bmod B_{ij} &\leq B_{ij} - d_i - d_j \\
\Leftrightarrow \left(\sum_{\ell \prec j} d_\ell - \sum_{\ell \prec i} d_\ell - d_i \right) \bmod B_{ij} &\leq B_{ij} - d_i - d_j \\
\Leftrightarrow \left(\sum_{i \preceq \ell \prec j} d_\ell - d_i \right) \bmod B_{ij} &\leq B_{ij} - d_i - d_j \\
\Leftrightarrow \left(\sum_{i \prec \ell \prec j} d_\ell \right) \bmod B_{ij} &\leq B_{ij} - d_i - d_j.
\end{aligned}$$

■

To show that the inclusion of zero-feasible orderings in the set of contiguous orderings is strict, we present the next example.

Example 3.2.8: Consider the set $\widetilde{\mathcal{P}}^5$ containing the four patterns $\widetilde{P}_1 = (24, 3, 1)$, $\widetilde{P}_2 = (24, 2, 1)$, $\widetilde{P}_3 = (24, 3, 1)$, and $\widetilde{P}_4 = (24, 4, 1)$. Define the order $\preceq$ by $1 \prec 2 \prec 3 \prec 4$. Now

$$\begin{aligned}
B_{12} &= \frac{24}{[3,2]} = \frac{24}{6} = 4 & \sum_{1\le\ell\le 2} d_\ell &= 2\\
B_{13} &= \frac{24}{[3,3]} = \frac{24}{3} = 8 & \sum_{1\le\ell\le 3} d_\ell &= 3\\
B_{14} &= \frac{24}{[3,4]} = \frac{24}{12} = 2 & \sum_{1\le\ell\le 4} d_\ell &= 4\\
B_{23} &= \frac{24}{[2,3]} = \frac{24}{6} = 4 & \sum_{2\le\ell\le 3} d_\ell &= 2\\
B_{24} &= \frac{24}{[2,4]} = \frac{24}{4} = 6 & \sum_{2\le\ell\le 4} d_\ell &= 3\\
B_{34} &= \frac{24}{[3,4]} = \frac{24}{12} = 2 & \sum_{3\le\ell\le 4} d_\ell &= 2
\end{aligned}$$

Thus $\preceq$ is not zero-feasible because $B_{14} \not\geq \sum_{1\le\ell\le 4} d_\ell$, but it is a contiguous ordering because

$$\left(\sum_{1<\ell<4} d_\ell\right) \bmod B_{14} = 2 \bmod 2 = 0$$

and

$$B_{14} - d_1 - d_4 = 2 - 1 - 1 = 0.$$

This implies that

$$\mathcal{T} = \left\{\widetilde{P}_1(0), \widetilde{P}_1(1), \widetilde{P}_2(2), \widetilde{P}_3(3)\right\}$$

is a feasible template.

Chapter 4

Complexity

In this chapter we establish the computational complexity of the problems discussed in Chapters 2-3. This will justify our work in Chapter 5 developing heuristics for these problems. We start with the problems involving linear patterns and their special orderings and then move on to proving that the general Feasible Fit Problem is NP-hard. Before we do that, however, we present a summary of complexity results from the periodic scheduling literature.

In the planar case, Vince [27] shows that maximizing the minimum distance between the vertices of regular polygons inscribed in a circle[1] is NP-complete by reduction from Graph Coloring[2]. In the linear case, Orlin [22] points out that constrained periodic assignment is NP-hard in the case of all periods being equal by reduction from Circular Arc Coloring. Korst, et al. [19] then extend the same reduction to the more general constrained periodic assignment problem. Later they show that the periodic scheduling problem is NP-complete [18] by reduction from Bin-Packing and is NP-complete in the strong sense even in the case of one processor [17] by reduction from 3-Partition. Bar-Noy, et al. [3] show that a different special case of periodic scheduling, where all processing times are unit length, is also NP-complete by reduction from Graph Coloring. Additionally, Serafini and Ukovich [26] show that the more general Periodic Event Scheduling Problem (PESP) is NP-complete by reduction from Hamiltonian Cycle.

[1] Please see Chapter 1 for more thorough definitions of the problems described here.

[2] See Garey & Johnson [12] for more information about common NP-complete problems (written in small caps) used in this section.

4.1 Linearized Fit Problems

We start this section by stating the decision problem versions of the three linearized problems defined in Section 3.1:

Linearized Feasible Fit:
Given a set of patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$, does there exist a set of starting positions S such that $\widetilde{\mathcal{P}}(S)$ is a feasible template?

Linearized Minimum Templates:
Given a set of patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ and $K \in \mathbb{N}$, does there exist a partition of I, $\mathcal{I} = \{I_u \mid u \in U\}$, with $\lvert U\rvert \leq K$, such that valid starting positions exist for each $\widetilde{\mathcal{P}}_u = \{\widetilde{P}_i \mid i \in I_u\}$?

Linearized Maximum Template:
Given a set of patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ and $K \in \mathbb{N}$, does there exist a subset I_1 of I, with $\lvert I_1\rvert \geq K$, such that valid starting positions exist for $\widetilde{\mathcal{P}}_1 = \{\widetilde{P}_i \mid i \in I_1\}$?

In Comment 3.1.3 we showed that the Periodic Scheduling Problem (PSP) from [17] is equivalent to the Linearized Minimum Templates Problem (LinMinTemp). Since [17] proves that Periodic Scheduling is NP-complete in the strong sense by reduction from 3-Partition, it follows that Linearized Minimum Templates is also NP-complete in the strong sense. Moreover, [17] reduces 3-Partition to an instance of Periodic Scheduling with only one processor. This shows that the special case of Linearized Minimum Templates where $K = 1$ (which is Linearized Feasible Fit) is also NP-complete in the strong sense. Finally, noting that Linearized Feasible Fit is a special case of Linearized Maximum Template where $K = |I|$ implies that the latter is also NP-complete in the strong sense.

4.1.1 Zero-Feasible Ordering Problem

We now determine the complexity of finding a zero-feasible ordering. First, let us define the Zero-Feasibility Problem (ZFP):

Zero-Feasibility:
Given a set of patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ where $\widetilde{P}_i = (B, n_i, d_i)$ for all $i \in I$, does there exist a total order $\preceq$ on I that is zero-feasible for $\mathcal{P}$?

We will prove that ZFP is NP-complete by reduction from the Bandwidth Problem:

BANDWIDTH:

Given an undirected graph $G = (N, A)$ and a positive integer $K \leq |N|$, does there exist a one-to-one function $\pi : N \to \{1, ..., |N|\}$ such that

$$|\pi(i) - \pi(j)| \leq K, \quad \forall \{i, j\} \in A?$$

Given an instance of the Bandwidth Problem ($G = (N, A)$ and $K \leq |N|$), we will construct a set of linear patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in N\}$ that will have a zero-feasible ordering $\preceq$ if and only if there is a permutation π that solves the given instance. First, let $d_i = 1$ for all $i \in N$. Moreover, we will connect the ordering $\preceq$ and the permutation π so that

$$\pi(j) - \pi(i) = \sum_{i \preceq \ell \preceq j} d_\ell - 1, \quad \forall i \preceq j. \tag{4.1.1}$$

We will choose B and n_i's such that

$$\sum_{i \preceq \ell \preceq j} d_\ell \leq B_{ij} \Rightarrow |\pi(j) - \pi(i)| \leq K, \quad \forall \{i, j\} \in A$$

and

$$\sum_{i \preceq \ell \preceq j} d_\ell \leq B_{ij}, \quad \forall \{i, j\} \notin A.$$

When $\{i, j\} \in A$, we will have for $i \prec j$,

$$\pi(j) - \pi(i) = \sum_{i \preceq \ell \preceq j} d_\ell - 1 \leq B_{ij} - 1.$$

Thus, $B_{ij} = K + 1$ if $\{i, j\} \in A$ and B_{ij} is greater than $|N| + 1$ if $\{i, j\} \notin A$. Since $B_{ij} = \dfrac{B}{[n_i, n_j]}$, choose the n_i's so that $[n_i, n_j] = \dfrac{B}{K+1}$ if $\{i, j\} \in A$ and $[n_i, n_j] < \dfrac{B}{|N|+1}$ if $\{i, j\} \notin A$. With this aim, define the following:

- Let Ω be the set of the first $\left|\left\{\{i, j\} \;\middle|\; \{i, j\} \notin A\right\}\right|$ primes.

- Associate with each pair $\{i,j\} \notin A$ a distinct prime, i.e.,

$$\phi : \Big\{ \{i,j\} \;\Big|\; \{i,j\} \notin A \Big\} \to \Omega.$$

- Let n_ℓ be the product of all primes in Ω that are associated with pairs $\{i,j\}$ such that $\ell \notin \{i,j\}$, raised to a sufficiently high power m, i.e.,

$$n_\ell = \prod_{\left\{\{i,j\} \middle| \{i,j\} \notin A, \ell \notin \{i,j\}\right\}} \phi\Big(\{i,j\}\Big)^m.$$

  In other words, n_ℓ is the product of all primes in Ω divided by the primes in Ω that are associated with arcs in $A^{\complement}$ incident with node ℓ, all raised to the power m. (The value of m will be determined later.)

- Let B be the product of all elements of Ω raised to the power m, and multiplied by $K+1$, i.e.,

$$B = (K+1) \prod_{\{i,j\} \notin A} \phi\big(\{i,j\}\big)^m.$$

We turn our attention to the value $[n_p, n_q]$. To determine the least common multiple of two integers a and b we factor them into their product of prime powers

$$a = p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r}$$
$$b = p_1^{b_1} p_2^{b_2} \cdots p_r^{b_r}$$

where some of the a_ℓ's and b_ℓ's could be zero. Then

$$[a,b] = p_1^{\max\{a_1,b_1\}} p_2^{\max\{a_2,b_2\}} \cdots p_r^{\max\{a_r,b_r\}}.$$

Now consider $[n_p, n_q]$. The prime factorizations of n_p and n_q are readily available since they are products of primes. Let

$$n_p = \prod_{\{i,j\} \notin A} \phi\big(\{i,j\}\big)^{a_{\{i,j\}}}$$
$$n_q = \prod_{\{i,j\} \notin A} \phi\big(\{i,j\}\big)^{b_{\{i,j\}}}.$$

Now, if $\{p,q\} \in A$, then for each $\{i,j\} \notin A$, at least one of p or q must not be in $\{i,j\}$ and $\max\{a_{\{i,j\}}, b_{\{i,j\}}\} = m$. Thus $[n_p, n_q]$ must be the product of all primes in Ω raised to the power m. Finally if $\{p,q\} \notin A$, then $a_{\{p,q\}} = b_{\{p,q\}} = 0$ and $\max\{a_{\{i,j\}}, b_{\{i,j\}}\} = m$ for all $\{i,j\} \neq \{p,q\}$. Thus,

$$[n_p, n_q] = \begin{cases} \displaystyle\prod_{\{i,j\} \notin A} \phi\Big(\{i,j\}\Big)^m, & \{p,q\} \in A \\ \displaystyle\frac{1}{\phi\Big(\{p,q\}\Big)^m} \prod_{\{i,j\} \notin A} \phi\Big(\{i,j\}\Big)^m, & \{p,q\} \notin A \end{cases}$$

and

$$B_{pq} = \begin{cases} K+1, & \{p,q\} \in A \\ (K+1) \cdot \phi\Big(\{p,q\}\Big)^m, & \{p,q\} \notin A. \end{cases}$$

Now, to ensure that $B_{pq} > |N| + 1$ for $\{p,q\} \notin A$, set $m = \left\lceil \log_2 \dfrac{|N|+1}{K+1} \right\rceil$, giving

$$\begin{aligned} B_{pq} &= (K+1) \cdot \phi\big(\{p,q\}\big)^{\left\lceil \log_2 \frac{|N|+1}{K+1} \right\rceil} \\ &> (K+1) \cdot 2^{\left\lceil \log_2 \frac{|N|+1}{K+1} \right\rceil} \\ &> (K+1) \cdot 2^{\log_2 \frac{|N|+1}{K+1}} \\ &= (K+1) \frac{|N|+1}{K+1} \\ &= |N| + 1. \end{aligned} \tag{4.1.2}$$

Finally then, we can state the following theorem.

Theorem 4.1.1 *The Zero-Feasibility Problem is NP-complete.*

Proof: First, (ZFP) is in NP since the inequalities defining zero-feasibility can be checked for a given order $\preceq$ in $O(|N|^2)$ time. Now let $G = (N, A)$ and $K \leq |N|$ be given. We will construct a set of linear patterns as follows. Let Ω be the set of the first $\left|\left\{\{i,j\} \;\middle|\; \{i,j\} \notin A\right\}\right|$ primes, let $\phi : \left\{\{i,j\} \;\middle|\; \{i,j\} \notin A\right\} \to \Omega$ be a one-to-one function (in other words there is a distinct prime for each edge in the complement of G). Now define the following values

$$
\begin{aligned}
d_i &:= 1, \quad \forall i \in N \\
m &:= \left\lceil \log_2 \frac{|N|+1}{K+1} \right\rceil \\
B &:= (K+1) \prod_{\{i,j\} \notin A} \phi\Big(\{i,j\}\Big)^m \\
a_{\ell,\{i,j\}} &:= \begin{cases} 0, & \ell \in \{i,j\}, \{i,j\} \notin A \\ m, & \text{else} \end{cases}, \quad \forall \ell, i \neq j \in N \\
n_\ell &:= \prod_{\{i,j\} \notin A} \phi\Big(\{i,j\}\Big)^{a_{\ell,\{i,j\}}}, \quad \forall \ell \in N.
\end{aligned}
$$

Finally, let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in N\}$ where $\widetilde{P}_i = (B, n_i, d_i)$. As discussed earlier,

$$
\begin{aligned}
[n_p, n_q] &= \prod_{\{i,j\} \notin A} \phi\Big(\{i,j\}\Big)^{\max\{a_{p,\{i,j\}}, a_{q,\{i,j\}}\}} \\
&= \begin{cases} \displaystyle\prod_{\{i,j\} \notin A} \phi\Big(\{i,j\}\Big)^m, & \{p,q\} \in A \\ \dfrac{1}{\phi\Big(\{p,q\}\Big)^m} \displaystyle\prod_{\{i,j\} \notin A} \phi\Big(\{i,j\}\Big)^m, & \{p,q\} \notin A \end{cases} \\
\Rightarrow B_{pq} &= \begin{cases} K+1, & \{p,q\} \in A \\ (K+1) \cdot \phi\Big(\{p,q\}\Big)^m, & \{p,q\} \notin A. \end{cases}
\end{aligned}
$$

It now remains to prove that there exists a function $\pi : N \to \{1, ..., |N|\}$ that solves the bandwidth problem for $G = (N, A)$ and K iff there exists an order $\preceq$ on N that is zero-feasible for $\mathcal{P}$.

($\Rightarrow$)

Given π define $\preceq$ such that

$$i \preceq j \Leftrightarrow \pi(i) \leq \pi(j), \quad \forall i, j \in N.$$

Suppose that π solves the Bandwidth Problem for $G = (N, A)$ and K. Then

$$|\pi(i) - \pi(j)| \leq K, \quad \forall \{i, j\} \in A.$$

Let $p \prec q \in N$ be given.

Case 1: $\{p, q\} \notin A$

Since $\pi(p), \pi(q) \leq |N|$ and all $d_\ell = 1$, relations (4.1.1) and (4.1.2) give

$$\begin{aligned} \sum_{p \preceq \ell \preceq q} d_\ell &= \pi(q) - \pi(p) + 1 \\ &\leq |N| + 1 \\ &\leq B_{pq}. \end{aligned}$$

Case 2: $\{p, q\} \in A$

Now $|\pi(q) - \pi(p)| \leq K$ and $B_{pq} = K + 1$, giving

$$\begin{aligned} \sum_{p \preceq \ell \preceq q} d_\ell &= \pi(q) - \pi(p) + 1 \\ &\leq |\pi(q) - \pi(p)| + 1 \\ &\leq K + 1 \\ &= B_{pq}. \end{aligned}$$

Thus $\preceq$ is zero-feasible for $\mathcal{P}$.

($\Leftarrow$)

Let $\preceq$ be zero-feasible for $\mathcal{P}$. Define $\pi : N \to \{1, ..., |N|\}$ according to

$$\pi(i) = \sum_{\ell \preceq i} 1, \quad \forall i \in N.$$

Let $\{p, q\} \in A$. Then

$$0 < \pi(q) - \pi(p) + 1 = \sum_{p \preceq \ell \preceq q} d_\ell \leq B_{pq} = K + 1.$$

Thus,

$$|\pi(q) - \pi(p)| \leq K$$

and π solves the bandwidth problem.

To ensure that this transformation takes polynomial time and space, first consider the time it takes to find the set Ω. As shown in [25], the mth prime number p_m is bounded by

$$p_m < m(\log m + \log\log m), \quad m \geq 6.$$

The size of Ω is

$$\left|\Big\{\{i,j\} \;\Big|\; \{i,j\} \notin A\Big\}\right| < \binom{|N|}{2} = O\Big(|N|^2\Big)$$

so to find the primes for Ω, we can run the Sieve of Eratosthenes on the set $\{x \in \mathbb{Z} \mid 2 \leq x \leq p_{|N|^2}\}$. This set has size

$$O\Big(|N|^2 \log |N|\Big).$$

Since the sieve runs in time $O(r \log\log r)$ on a set of size r it will take

$$\begin{aligned} O\Big(|N|^2 \log |N| \log\log\big(|N|^2 \log |N|\big)\Big) &= O\Big(|N|^2 \log |N| \big(\log\log |N|^2 + \log\log\log |N|\big)\Big) \\ &= O\Big(|N|^2 \log |N| \log\log |N|\Big) \end{aligned}$$

time to find Ω.

Finally, we need to consider the encoding length of B and the n_i's. Since $B > n_i$ for all $i \in N$, we just need to consider the length of B:

$$\begin{aligned}
\log B &= \log\left((K+1)\prod_{\{i,j\}\notin A}\phi\big(\{i,j\}\big)^{\left\lceil \log_2 \frac{|N|+1}{K+1}\right\rceil}\right)\\
&= \log(K+1) + \sum_{\{i,j\}\notin A}\log\left(\phi\big(\{i,j\}\big)^{\left\lceil \log_2 \frac{|N|+1}{K+1}\right\rceil}\right)\\
&= \log(K+1) + \left\lceil \log_2 \frac{|N|+1}{K+1}\right\rceil \sum_{\{i,j\}\notin A}\log\big(\phi\big(\{i,j\}\big)\big)\\
&= O\big(\log|N|\big) + O\big(\log|N|\big)\sum_{\{i,j\}\notin A}\log\Big(O\big(|N|^2\log|N|\big)\Big)\\
&= O\big(\log|N|\big) + O\big(\log|N|\big)O\big(|N|^2\big)\log\Big(O\big(|N|^2\log|N|\big)\Big)\\
&= O\big(\log|N|\big) + O\big(\log|N|\big)O\big(|N|^2\big)O\big(\log|N|^2 + \log\log|N|\big)\\
&= O\big(\log|N|\big) + O\big(\log|N|\big)O\big(|N|^2\big)O\big(\log|N|\big)\\
&= O\big(|N|^2\log^2|N|\big).
\end{aligned}$$

■

4.2 Generalized Fit Problem

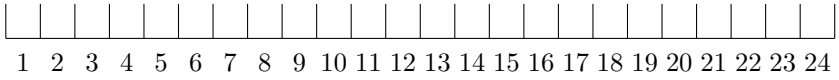
Having shown the complexity of the problems associated with linearized patterns $\widetilde{P_i} = (B, n_i, d_i)$, we return to the problems associated with generalized patterns $P_i = (R_i, n_i, \rho_i)$. Bar-Noy, et al. [3] discuss a special case of the Periodic Scheduling Problem with only one server and $e(i) = 1$ for all i which they call the Periodic Maintenance Scheduling Problem (PMSP):

Periodic Maintenance Scheduling:
Given m machines and service intervals $l_1, l_2, ..., l_m$ such that $\sum_{i=1}^{m}\frac{1}{l_i} \leq 1$, does there exist an infinite maintenance service schedule of these machines for which consecutive services of machine i are exactly l_i time slots apart and no more than one machine is serviced in a single time slot?

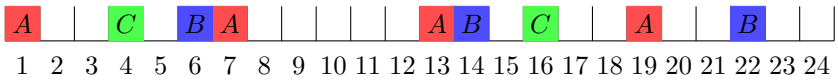
They show that PMSP is NP-complete by reduction from Graph Coloring. We wish to show that FFP($\mathcal{P}$) is NP-hard by reduction from PMSP. To do this, given any specific instance of PMSP,

we need to construct a set of patterns that can be fit on a single feasible template iff the instance of PMSP has a feasible maintenance schedule. More importantly, we need to show how a set of starting positions that are valid for our constructed set of patterns can be transformed into a feasible maintenance schedule for a PMSP instance. We illustrate our approach in the following example.

Consider an instance of PMSP with machines $\{A, B, C\}$ and service intervals $(l_A, l_B, l_C) = (6, 8, 12)$. Each machine takes a full day to process, so building a schedule can be seen as filling the days on a calendar. The days are viewed as empty bins as shown in Figure 4.2.1(a). Note that, since $\text{lcm}(6, 8, 12) = 24$, the schedule for the first 24 days yields the exact same schedule for the next 24 days and every 24 day period after that [18]. So, for example, scheduling the maintenance on machine A for day 1 fills bins $1 + 6k$ for $k \in \mathbb{Z}$. Similarly, we can schedule B and C, as shown in Figure 4.2.1(b).



4.2.1(a): Bins for a PMSP example

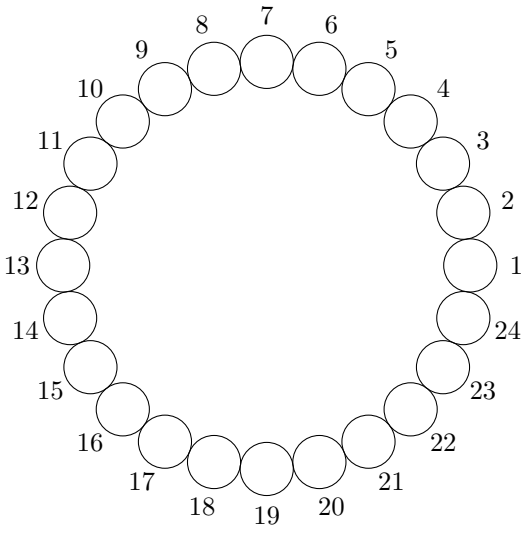


4.2.1(b): A valid PMSP schedule

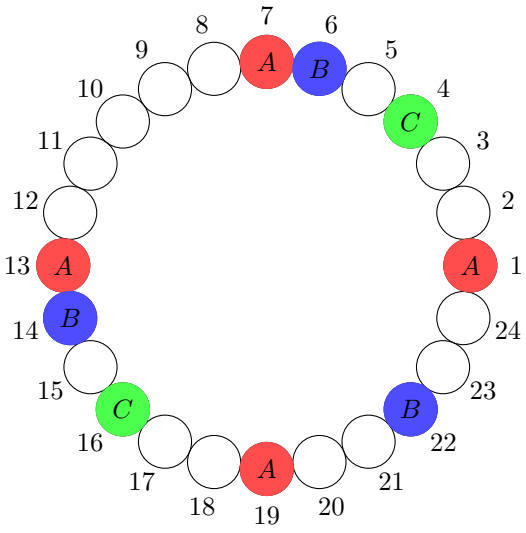
Figure 4.2.1: PMSP Example

Instead of thinking of the days as intervals on the number line, consider them as disks of diameter 1, arranged around the circumference of a circle as shown in Figure 4.2.2(a). Then maintenance can be scheduled by filling in the disks as in Figure 4.2.2(b). The periodic maintenance of a machine is now a pattern of disks as defined in Section 2.1.

It is useful to note that the centers of the disks in Figure 4.2.2(a) define the vertices of a regular 24-gon whose sides are of length 1. Thus the radius of the outer circle can be calculated as $\frac{1}{2}\csc\frac{\pi}{24}$. The disks themselves have radii of 0.5 and for each machine the corresponding pattern has $n_i = 24/l_i$ disks. The main problem, however, is that valid starting positions for the patterns can be real-valued, but a feasible schedule for PMSP must be integer. If the starting angles for the



4.2.2(a): PMSP bins arranged as disks



4.2.2(b): A valid PMSP schedule as disks

Figure 4.2.2: PMSP Example: Transformation to disks

patterns place the centers of the disks exactly on the vertices of the regular 24-gon, then the starting angles will be of the form $\varphi_i = \frac{2\pi}{24} v_i$ where $v_i \in \mathbb{Z}$. In this case, the v_i's form a feasible maintenance schedule. The following lemma generalizes this construction.

Lemma 4.2.1 *Let* $\mathcal{P} = \{P_i \mid i \in I\}$ *be given where* $P_i = \left(\frac{1}{2}\csc\frac{\pi}{L}, n_i, \frac{1}{2}\right)$ *and* $n_i \mid L$ *for all* $i \in I$. *Then* $\mathcal{S}(\mathcal{P}) \neq \emptyset$ *iff there exist* $\{\hat{v}_i \in \mathbb{Z} \mid i \in I\}$ *such that* $\left\{\hat{\varphi}_i = \frac{2\pi}{L}\hat{v}_i \middle| i \in I\right\} \in \mathcal{S}(\mathcal{P})$.

That is, if there are valid starting positions, then there must be valid starting positions that lie on the points of the regular L-gon centered at the origin with sides of length 1.

Proof:

($\Leftarrow$)

If there exists $\{\hat{v}_i \in \mathbb{Z} \mid i \in I\}$ such that $\left\{\hat{\varphi}_i = \frac{2\pi}{L}\hat{v}_i \middle| i \in I\right\} \in \mathcal{S}(\mathcal{P})$ then $\mathcal{S}(\mathcal{P})$ cannot be empty.

($\Rightarrow$)

Let $\mathcal{P} = \{P_i \mid i \in I\}$ be given as above and let $\preceq$ be a total order on I. Assume that $\{P_i(\overline{\varphi}_i) \mid i \in I\}$ is a feasible template. First, we calculate the following values[3]:

$$\beta_{ij} = \frac{2\pi}{[n_i, n_j]}$$
$$\delta_{ij} = 2\arcsin\left(\frac{\rho_i + \rho_j}{2R}\right) = 2\arcsin\left(\frac{1}{\csc\frac{\pi}{L}}\right) = 2\arcsin\left(\sin\frac{\pi}{L}\right) = \frac{2\pi}{L}.$$

Define $\overline{v}_i = \frac{L}{2\pi}\overline{\varphi}_i$ for all $i \in I$. Now we know from Theorem 2.3.1 that there exists $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$ such that for $i \prec j$,

$$\begin{aligned} k_{ij}\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k_{ij}+1)\beta_{ij} - \delta_{ij} \\ k_{ij}\frac{2\pi}{[n_i,n_j]} + \frac{2\pi}{L} \leq \frac{2\pi}{L}\overline{v}_j - \frac{2\pi}{L}\overline{v}_i \leq (k_{ij}+1)\frac{2\pi}{[n_i,n_j]} - \frac{2\pi}{L} \\ k_{ij}\frac{L}{[n_i,n_j]} + 1 \leq \overline{v}_j - \overline{v}_i \leq (k_{ij}+1)\frac{L}{[n_i,n_j]} - 1. \end{aligned} \tag{4.2.1}$$

Consider the network $\breve{G} = (N, A, \breve{w})$ where

$$N = I$$
$$A = \{(i,j) \mid i \neq j \in I\}$$
$$\breve{w}_{ij} = \begin{cases} (k_{ij}+1)\dfrac{L}{[n_i,n_j]} - 1, & i \prec j \\ -k_{ji}\dfrac{L}{[n_j,n_i]} - 1, & j \prec i. \end{cases}$$

[3]Recall that, since all patterns are arranged on the same outer circumference, we can use the simplified formula for δ_{ij} from Corollary 2.2.16.

Note that $\breve{G}$ is the same network as $G(\mathcal{P}, \preceq, K)$ except that $\breve{w}_{ij} = \dfrac{L}{2\pi} w_{ij}$. Let C be any cycle in $\breve{G}$ and define $C_{\prec}$ and $C_{\succ}$ as in (2.4.2). Then by (4.2.1)

$$\begin{aligned}
\breve{w}[C] &= \sum_{(i,j)\in C} \breve{w}_{ij} \\
&= \sum_{(i,j)\in C_{\prec}} \left[(k_{ij}+1)\frac{L}{[n_i, n_j]} - 1 \right] - \sum_{(i,j)\in C_{\succ}} \left[k_{ji}\frac{L}{[n_j, n_i]} + 1 \right] \\
&\geq \sum_{(i,j)\in C_{\prec}} (\overline{v}_j - \overline{v}_i) + \sum_{(i,j)\in C_{\succ}} (\overline{v}_j - \overline{v}_i) \\
&= \sum_{(i,j)\in C} (\overline{v}_j - \overline{v}_i) \\
&= 0.
\end{aligned}$$

Since our selection of C was arbitrary, $\breve{G}$ must have no negative cycles. Next notice that, since $[n_i, n_j] \mid L$ for all i, j, $\breve{w}_{ij} \in \mathbb{Z}$ for all (i, j). Thus the shortest path distances $\hat{v}_i$ from some given root node $r \in I$ must be integer and these must also satisfy for all $i \prec j$,

$$\begin{gathered}
k_{ij}\frac{L}{[n_i, n_j]} + 1 \leq \hat{v}_j - \hat{v}_i \leq (k_{ij}+1)\frac{L}{[n_i, n_j]} - 1 \\
k_{ij}\frac{2\pi}{[n_i, n_j]} + \frac{2\pi}{L} \leq \frac{2\pi}{L}\hat{v}_j - \frac{2\pi}{L}\hat{v}_i \leq (k_{ij}+1)\frac{2\pi}{[n_i, n_j]} - \frac{2\pi}{L} \\
k_{ij}\beta_{ij} + \delta_{ij} \leq \hat{\varphi}_j - \hat{\varphi}_i \leq (k_{ij}+1)\beta_{ij} - \delta_{ij}
\end{gathered}$$

where $\hat{\varphi}_i = \dfrac{2\pi}{L}\hat{v}_i$ for all $i \in I$. ■

The lemma above shows that, with a properly constructed set of patterns, we can build a special network $\breve{G}$ that can be used to find starting positions that are multiples of $\dfrac{2\pi}{L}$. Unfortunately, building the set of patterns detailed in Lemma 4.2.1 is not practical since in general we cannot calculate $R = \dfrac{1}{2}\csc\dfrac{\pi}{L}$ precisely in polynomial time (in fact, if R is irrational, we may not be able to calculate it precisely in any finite amount of time). To remedy this we next prove that, even if R is calculated to within some tolerance of the value defined in Lemma 4.2.1, the same solution structure exists.

Lemma 4.2.2 *Let* $\mathcal{P} = \{P_i \mid i \in I\}$ *be given where* $P_i = \left(\frac{1}{2}\csc\frac{\pi}{L}, n_i, \frac{1}{2}\right)$ *and* $n_i \mid L$ *for all* $i \in I$. *Define* $\mathcal{P}' = \{P'_i \mid i \in I\}$ *where* $P'_i = \left(\frac{1}{2}\csc\frac{\pi}{L} + \varepsilon, n_i, \frac{1}{2}\right)$. *Then* $\mathcal{S}(\mathcal{P}) \neq \emptyset$ *iff* $\mathcal{S}(\mathcal{P}') \neq \emptyset$ *for all* $0 \leq \varepsilon < \dfrac{1}{16|I|^2}$.

Proof: Throughout this proof we refer to the values associated with the set $\mathcal{P}'$ with primes (i.e., β'_{ij}, δ'_{ij}, and w'_{ij}). The values associated with the set $\mathcal{P}$ will remain unadorned (i.e., β_{ij}, δ_{ij}, and w_{ij}). It should be noted at this point, however, that since n_i occurs in both P_i and P'_i, $\beta_{ij} = \beta'_{ij}$ for all $i, j \in I$. Recall from the proof of Lemma 4.2.1 that

$$\delta_{ij} = \frac{2\pi}{L}.$$

Before we proceed to the actual proof, let us bound how much the error ε in the value of R' affects the calculation of δ'_{ij}. First, note that

$$\varepsilon < \frac{1}{16|I|^2} < \frac{\pi^2}{16|I|^2}.$$

Thus

$$2\varepsilon < \frac{1}{2} \cdot \frac{\pi^2}{4|I|^2} = \frac{1}{2}\left(\frac{\pi}{2|I|}\right)^2. \tag{4.2.2}$$

Next, recall the Taylor series expansion for $\sec x$:

$$\sec x = \sum_{n=0}^{\infty} \frac{|E_{2n}|x^{2n}}{(2n)!} = 1 + \frac{1}{2}x^2 + \sum_{n=2}^{\infty} \frac{|E_{2n}|x^{2n}}{(2n)!}$$

where $|E_{2n}|$ are the Euler numbers[4]. Since all terms in the summation are positive for $x > 0$,

$$\sec x > 1 + \frac{1}{2}x^2, \quad 0 < x < \frac{\pi}{2}.$$

[4]The Online Encyclopedia of Integer Sequences (http://oeis.org) #A000364

Thus, continuing from (4.2.2),

$$\begin{aligned}
2\varepsilon &< \frac{1}{2}\left(\frac{\pi}{2|I|}\right)^2 \\
&< \sec\frac{\pi}{2|I|} - 1 \\
&= \csc\left(\frac{\pi}{2} - \frac{\pi}{2|I|}\right) - \csc\frac{\pi}{2} \\
&\leq \csc\left(\frac{\pi}{L} - \frac{\pi}{L|I|}\right) - \csc\frac{\pi}{L}, \quad \forall L \geq 2.
\end{aligned}$$

Finally,

$$\begin{aligned}
2\varepsilon &< \csc\left(\frac{\pi}{L} - \frac{\pi}{L|I|}\right) - \csc\frac{\pi}{L} \\
\csc\frac{\pi}{L} + 2\varepsilon &< \csc\left(\frac{\pi}{L} - \frac{\pi}{L|I|}\right) \\
\frac{1}{\csc\frac{\pi}{L} + 2\varepsilon} &> \sin\left(\frac{\pi}{L} - \frac{\pi}{L|I|}\right) \\
\arcsin\left(\frac{1}{\csc\frac{\pi}{L} + 2\varepsilon}\right) &> \frac{\pi}{L} - \frac{\pi}{L|I|} \\
\arcsin\left(\frac{1}{2\frac{1}{2}\csc\frac{\pi}{L} + 2\varepsilon}\right) &> \arcsin\left(\sin\left(\frac{\pi}{L}\right)\right) - \frac{\pi}{L|I|} \\
\arcsin\left(\frac{1}{2(\frac{1}{2}\csc\frac{\pi}{L} + \varepsilon)}\right) &> \arcsin\left(\frac{1}{\csc\frac{\pi}{L}}\right) - \frac{\pi}{L|I|} \\
\arcsin\left(\frac{1}{2(\frac{1}{2}\csc\frac{\pi}{L} + \varepsilon)}\right) &> \arcsin\left(\frac{1}{2(\frac{1}{2}\csc\frac{\pi}{L})}\right) - \frac{\pi}{L|I|} \\
2\arcsin\left(\frac{1}{2R'}\right) &> 2\arcsin\left(\frac{1}{2R}\right) - \frac{2\pi}{L|I|} \\
2\arcsin\left(\frac{1}{2R}\right) - 2\arcsin\left(\frac{1}{2R'}\right) &< \frac{2\pi}{L|I|} \\
\delta_{ij} - \delta'_{ij} &< \frac{2\pi}{L|I|}. \qquad (4.2.3)
\end{aligned}$$

To bound the error in the other direction, note that since $\varepsilon \geq 0$,

$$\begin{aligned}
\frac{1}{2}\csc\frac{\pi}{L} + \varepsilon &\geq \frac{1}{2}\csc\frac{\pi}{L} \\
2\left(\frac{1}{2}\csc\frac{\pi}{L} + \varepsilon\right) &\geq 2\left(\frac{1}{2}\csc\frac{\pi}{L}\right) \\
\frac{1}{2(\frac{1}{2}\csc\frac{\pi}{L} + \varepsilon)} &\leq \frac{1}{2(\frac{1}{2}\csc\frac{\pi}{L})} \\
\arcsin\left(\frac{1}{2(\frac{1}{2}\csc\frac{\pi}{L} + \varepsilon)}\right) &\leq \arcsin\left(\frac{1}{2(\frac{1}{2}\csc\frac{\pi}{L})}\right) \\
0 &\leq 2\arcsin\left(\frac{1}{2R}\right) - 2\arcsin\left(\frac{1}{2R'}\right) \\
0 &\leq \delta_{ij} - \delta'_{ij}.
\end{aligned}$$

Combining this with (4.2.3), we get

$$0 \leq \delta_{ij} - \delta'_{ij} < \frac{2\pi}{L|I|}. \tag{4.2.4}$$

We continue, showing that $\mathcal{S}(\mathcal{P}) \neq \emptyset$ iff $\mathcal{S}(\mathcal{P}') \neq \emptyset$.

($\Rightarrow$)

Assume that $\mathcal{S}(\mathcal{P}) \neq \emptyset$. Let the total order $\preceq$ on I be given. Then there exists $\varphi_i \in \mathbb{R}$ for all $i \in I$ and $k_{ij} \in \mathbb{Z}$ for all $i \prec j$ such that

$$k_{ij}\beta_{ij} + \delta_{ij} \leq \varphi_j - \varphi_i \leq (k_{ij}+1)\beta_{ij} - \delta_{ij}.$$

Since $0 \leq \delta_{ij} - \delta'_{ij}$, we have $\delta_{ij} \geq \delta'_{ij}$. Thus,

$$k_{ij}\beta'_{ij} + \delta'_{ij} \leq k_{ij}\beta_{ij} + \delta_{ij} \leq \varphi_j - \varphi_i \leq (k_{ij}+1)\beta_{ij} - \delta_{ij} \leq (k_{ij}+1)\beta'_{ij} - \delta'_{ij},$$

and so the φ_i's are valid for $\mathcal{P}'$.

($\Leftarrow$)

Assume that $\mathcal{S}(\mathcal{P}') \neq \emptyset$ and let the total order $\preceq$ on I be given. Now, let $\overline{K} = \{\overline{k}_{ij} \mid i \prec j\}$ be given such that $G(\mathcal{P}', \preceq, \overline{K})$ has no negative cycles. Such a $\overline{K}$ exists by Theorem 2.4.5. We prove that $G(\mathcal{P}, \preceq, \overline{K})$ has no negative cycles.

Let C be any directed cycle in $G(\mathcal{P}', \preceq, \overline{K})$. Let $C_\prec$ and $C_\succ$ be defined as in (2.4.2). Then

$$\begin{aligned}
0 &\leq w'[C] \\
&= \sum_{(i,j)\in C_\prec} (\overline{k}_{ij}+1)\beta'_{ij} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\beta'_{ji} - \sum_{(i,j)\in C} \delta'_{ij} \\
&< \sum_{(i,j)\in C_\prec} (\overline{k}_{ij}+1)\beta_{ij} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\beta_{ji} - \sum_{(i,j)\in C} \left(\delta_{ij} - \frac{2\pi}{L|I|}\right)
\end{aligned}$$

by (4.2.4). Continuing,

$$\begin{aligned}
0 &< \sum_{(i,j)\in C_\prec} (\overline{k}_{ij}+1)\beta_{ij} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\beta_{ji} - \sum_{(i,j)\in C} \delta_{ij} + \sum_{(i,j)\in C} \frac{2\pi}{L|I|} \\
0 &< \sum_{(i,j)\in C_\prec} (\overline{k}_{ij}+1)\frac{2\pi}{[n_i,n_j]} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\frac{2\pi}{[n_j,n_i]} - \sum_{(i,j)\in C} \frac{2\pi}{L} + \sum_{(i,j)\in C} \frac{2\pi}{L|I|} \\
0 &< \sum_{(i,j)\in C_\prec} (\overline{k}_{ij}+1)\frac{L}{[n_i,n_j]} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\frac{L}{[n_j,n_i]} - \sum_{(i,j)\in C} 1 + \sum_{(i,j)\in C} \frac{1}{|I|} \\
-\sum_{(i,j)\in C} \frac{1}{|I|} &< \sum_{(i,j)\in C_\prec} (\overline{k}_{ij}+1)\frac{L}{[n_i,n_j]} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\frac{L}{[n_j,n_i]} - \sum_{(i,j)\in C} 1.
\end{aligned}$$

Now, since $|C| \leq |I|$,

$$\begin{aligned}
-\sum_{(i,j)\in C} \frac{1}{|C|} &< \sum_{(i,j)\in C_\prec} (\overline{k}_{ij}+1)\frac{L}{[n_i,n_j]} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\frac{L}{[n_j,n_i]} - \sum_{(i,j)\in C} 1 \\
-1 &< \sum_{(i,j)\in C_\prec} (\overline{k}_{ij}+1)\frac{L}{[n_i,n_j]} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\frac{L}{[n_j,n_i]} - \sum_{(i,j)\in C} 1.
\end{aligned}$$

Note that, since $[n_i, n_j] \mid L$ for all i, j, all terms of the right-hand side of this inequality are integer. Thus

$$\begin{aligned}
0 &\leq \sum_{(i,j)\in C_\prec} (\overline{k}_{ij} + 1)\frac{L}{[n_i, n_j]} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\frac{L}{[n_j, n_i]} - \sum_{(i,j)\in C} 1 \\
0 &\leq \sum_{(i,j)\in C_\prec} (\overline{k}_{ij} + 1)\frac{2\pi}{[n_i, n_j]} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\frac{2\pi}{[n_j, n_i]} - \sum_{(i,j)\in C} \frac{2\pi}{L} \\
0 &\leq \sum_{(i,j)\in C_\prec} (\overline{k}_{ij} + 1)\beta_{ij} - \sum_{(i,j)\in C_\succ} \overline{k}_{ji}\beta_{ji} - \sum_{(i,j)\in C} \delta_{ij} \\
0 &\leq w[C].
\end{aligned}$$

Our choice of C was arbitrary and so $G(\mathcal{P}, \preceq, \overline{K})$ has no negative cycles. This implies by Theorem 2.4.5 that $\mathcal{S}(\mathcal{P}) \neq \emptyset$. ■

Now, combining these two lemmas, we can say that if $\mathcal{P}'$ has valid starting positions, then $\mathcal{P}$ has valid starting positions that are multiples of $\frac{2\pi}{L}$. We should also note that, as shown in the proof of Lemma 4.2.2, any valid starting positions for $\mathcal{P}$ are also valid for $\mathcal{P}'$. This will enable us to create a set of patterns that can be used to find a solution to any instance of PMSP. Before the main theorem, however, we need to state one more lemma, this one regarding PMSP.

Lemma 4.2.3 [5]**(Bar-Noy, Bhatia, Naor, & Schieber [3])**
A given integer vector $\vec{v}$ determines a feasible schedule for the PMSP instance, if and only if, the following holds for any pair of machines i and j:

$$v_i - v_j \neq 0 \bmod (l_i, l_j).$$

Finally, we can prove the following:

Theorem 4.2.4 *The generalized Feasible Fit Problem is NP-Hard.*

Proof: Let a number of machines m and service intervals $l_1, l_2, ..., l_m$ be given. Without loss of generality, let $m \geq 3$ (Recall that the case with $m = 2$ can be solved immediately as detailed in Section 2.2). We will construct a set of patterns which can be fit on one template iff a valid

[5] *cf.* Corollary 3.1.4

maintenance service schedule exists. We construct our patterns by defining the following:

$$\begin{aligned} I &:= \{1, 2, 3, ..., m\} \\ L &:= \text{lcm}(l_1, l_2, ..., l_m) = [l_1, l_2, ..., l_m] \\ R &:= \frac{1}{2} \csc \frac{\pi}{L} \\ \rho &:= \frac{1}{2} \\ n_i &:= \frac{L}{l_i}, \quad \forall i \in I. \end{aligned}$$

We also define $R' := R + \varepsilon$ for some $0 \leq \varepsilon < \dfrac{1}{16m^2}$. Note that since $L = n_i \cdot l_i$, we have $n_i \mid L$ for all $i \in I$ and

$$(l_i, l_j) = \left(\frac{L}{n_i}, \frac{L}{n_j}\right) = \frac{L}{[n_i, n_j]}.$$

We define the sets of patterns $\mathcal{P} = \left\{P_i = (R, n_i, \rho) \;\middle|\; i \in I\right\}$ and $\mathcal{P}' = \left\{P'_i = (R', n_i, \rho) \;\middle|\; i \in I\right\}$ and claim that the given instance of PMSP has a solution iff $\mathcal{S}(\mathcal{P}') \neq \emptyset$. More importantly, we detail how valid starting positions for $\mathcal{P}'$ can be used to find a valid maintenance service schedule in polynomial time.

Recall from the proof of Lemma 4.2.2 that

$$\delta_{ij} = \frac{2\pi}{L}, \qquad\qquad \beta_{ij} = \beta'_{ij} = \frac{2\pi}{[n_i, n_j]},$$

and

$$0 \leq \delta_{ij} - \delta'_{ij} < \frac{2\pi}{L|I|} = \frac{2\pi}{Lm}.$$

($\Rightarrow$)

Let $\vec{v}$ be a valid maintenance service schedule. Lemma 4.2.3 ensures that for $1 \leq i \neq j \leq m$

$$v_i - v_j \neq 0 \bmod (l_i, l_j).$$

Note that, since $\vec{v}$ is integer, this is equivalent to

$$1 \leq (v_j - v_i) \bmod (l_i, l_j) \leq (l_i, l_j) - 1$$
$$1 \leq v_j - v_i - k_{ij}(l_i, l_j) \leq (l_i, l_j) - 1$$

for some $k_{ij} \in \mathbb{Z}$. Continuing,

$$\begin{aligned}
1 \leq v_j - v_i - k_{ij}(l_i, l_j) &\leq (l_i, l_j) - 1 \\
1 \leq v_j - v_i - k_{ij}\frac{L}{[n_i, n_j]} &\leq \frac{L}{[n_i, n_j]} - 1 \\
k_{ij}\frac{L}{[n_i, n_j]} + 1 \leq v_j - v_i &\leq (k_{ij} + 1)\frac{L}{[n_i, n_j]} - 1 \\
k_{ij}\frac{2\pi}{[n_i, n_j]} + \frac{2\pi}{L} \leq \frac{2\pi}{L}v_j - \frac{2\pi}{L}v_i &\leq (k_{ij} + 1)\frac{2\pi}{[n_i, n_j]} - \frac{2\pi}{L} \\
k_{ij}\beta_{ij} + \delta_{ij} \leq \frac{2\pi}{L}v_j - \frac{2\pi}{L}v_i &\leq (k_{ij} + 1)\beta_{ij} - \delta_{ij}.
\end{aligned}$$

Now, since $\delta'_{ij} \leq \delta_{ij}$,

$$k_{ij}\beta'_{ij} + \delta'_{ij} \leq k_{ij}\beta_{ij} + \delta_{ij} \leq \frac{2\pi}{L}v_j - \frac{2\pi}{L}v_i \leq (k_{ij} + 1)\beta_{ij} - \delta_{ij} \leq (k_{ij} + 1)\beta'_{ij} - \delta'_{ij}.$$

Letting $\hat{\varphi}_i = \dfrac{2\pi}{L}v_i$ for all $i \in I$, we get

$$k_{ij}\beta'_{ij} + \delta'_{ij} \leq \hat{\varphi}_j - \hat{\varphi}_i \leq (k_{ij} + 1)\beta'_{ij} - \delta'_{ij}.$$

Thus $\{\hat{\varphi}_i \mid i \in I\} \in \mathcal{S}(\mathcal{P}')$ and $\mathcal{S}(\mathcal{P}') \neq \emptyset$.

($\Leftarrow$)

Assume that $\{P_i'(\overline{\varphi}_i) \mid i \in I\}$ is a feasible template. Let the total order $\preceq$ on I be given. Lemmas 4.2.1 and 4.2.2 imply that there is a set of starting positions of the form $\hat{\varphi}_i = \frac{2\pi}{L} v_i$ where $v_i \in \mathbb{Z}$, but we still need to show how to find those solutions.

We start by using (2.2.5) to calculate the set $\overline{K}$ from the valid starting positions $\overline{\varphi}_i$. Since the resulting $G(\mathcal{P}', \preceq, \overline{K})$ has no negative cycles, we know that $G(\mathcal{P}, \preceq, \overline{K})$ has no negative cycles by Lemma 4.2.2. Following the construction used in the proof of Lemma 4.2.1 we build network $\breve{G}$ which we know also has no negative cycles. Finally, we find shortest path distances v_i from some given root node r in $\breve{G}$. Because of the integrality of the weights of $\breve{G}$, these v_i are integer for all $i \in I$ and, since they are shortest path distances in $\breve{G}$, satisfy for $i \prec j$

$$
\begin{gathered}
-\breve{w}_{ji} \leq v_j - v_i \leq \breve{w}_{ij} \\
k_{ij} \frac{L}{[n_i, n_j]} + 1 \leq v_j - v_i \leq (k_{ij} + 1) \frac{L}{[n_i, n_j]} - 1 \\
1 \leq v_j - v_i - k_{ij} \frac{L}{[n_i, n_j]} \leq \frac{L}{[n_i, n_j]} - 1 \\
1 \leq v_j - v_i - k_{ij}(l_i, l_j) \leq (l_i, l_j) - 1 \\
1 \leq (v_j - v_i) \bmod (l_i, l_j) \leq (l_i, l_j) - 1 \\
v_j - v_i \neq (0 \bmod (l_i, l_j)).
\end{gathered}
$$

Thus, $\vec{v}$ is a valid maintenance schedule for the instance of PMSP.

Finally, we need to ensure that this transformation does in fact take polynomial time and space. Note that we can calculate $\csc x$ to n bits of precision in $O(n^{2.5})$ time using Taylor series. Thus, if we calculate R to at least $5 \log_2 m$ bits of precision (which we can do in $O(\log^{2.5} m)$ time) then

$$\begin{aligned} R' - R &< \frac{1}{2^{5 \log_2 m}} \\ &= \frac{1}{m^5} \\ &\leq \frac{1}{3^3 \cdot m^2} \\ &= \frac{1}{27m^2} \\ &< \frac{1}{16m^2} \\ &= \frac{1}{16|I|^2} \end{aligned}$$

as required. To ensure that L is of reasonable size, let $l = \max_{i=1,\ldots,m} \{l_i\}$, then

$$\begin{aligned} L &\leq \prod_{i=1}^{m} l_i \\ \Rightarrow \log L &\leq \log \prod_{i=1}^{m} l_i \\ &= \sum_{i=1}^{m} \log l_i \\ &\leq \sum_{i=1}^{m} \log l \\ &= m \log l. \end{aligned}$$

Finally, the Euclidean Algorithm for computing the greatest common divisor of two integers runs in time $O(\log^2 n)$ [9] where n is the number of bits in the smaller of the two integers. It can also be modified to find the least common multiple in the same asymptotic time. Thus, calculating L requires repeatedly calculating $\left[[l_1, l_2, \ldots, l_k], l_{k+1}\right]$ which takes time $O(\log^2 \log l)$. As a result, the total time required to calculate L is $O(m \log^2 \log l)$. ■

Chapter 5

Heuristics

In Chapter 4 we showed that finding feasible templates is intrinsically difficult and thus finding exact answers to the problems studied in this dissertation is impractical in most cases. As a result, we develop heuristics for the solution of all three posed problems. We start by exploiting the special orderings described in Section 3.2 to solve problems involving linearized patterns. We then present an additional heuristic based on the network formulation presented in Section 2.4. This is first implemented for linearized patterns but is then extended to generalized patterns. To begin, we consider the problem of trying to find zero-feasible orderings.

5.1 Ordering-Based Heuristics

Throughout this section we will use the value $B_{ij} - d_i - d_j$ often and therefore define

$$B'_{ij} := B_{ij} - d_i - d_j = \frac{B}{[n_i, n_j]} - d_i - d_j.$$

We now discuss two heuristics for building feasible solutions to the proposed problems. They are both based on the zero-feasible inequalities discussed in Section 3.2 and are thus only valid for linearized patterns. Both grow a zero-feasible solution by adding patterns to an existing zero-feasible solution. We know that zero-feasibility is based on the given total order $\preceq$ on the patterns. The inequality

$$B'_{ij} \geq \sum_{i \prec \ell \prec j} d_\ell, \quad \forall i \prec j \tag{5.1.1}$$

is equivalent to Definition 3.2.3. Our convention is that, if the range of the summation $\sum_{i \prec \ell \prec j} d_\ell$ is empty, then its value is zero. Relation (5.1.1) requires that pairs (i, j) that are adjacent with respect to $\preceq$ have non-negative B'_{ij}. On the other hand, pairs that are further apart with respect to $\preceq$, are required to have B'_{ij} not less than the sum of the d_ℓ's strictly between i and j in the given order. To this end, we would like to arrange the patterns on a template in such a way that those pairs with large B'_{ij} are far apart and, more importantly, that those pairs with small but positive B'_{ij} are kept close together in the ordering. Thus, if we consider B'_{ij}'s for all (i, j) pairs, it is desirable to first link together those pairs with smallest positive B'_{ij} before possibly considering those pairs with larger B'_{ij}. In this way these heuristics closely resemble classical algorithms for constructing a minimum spanning tree (MST) [1].

We first establish some notation. Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ be the given set of patterns. Let $I_u \subseteq I$ and define $\widetilde{\mathcal{P}}_u = \{\widetilde{P}_i \mid i \in I_u\}$. Let $\preceq_u$ be a strict total order on I_u and let

$$s^*_i = \sum_{\ell \prec_u i} d_\ell \quad \forall i \in I_u$$

be defined as in (3.2.5). Our objective is to build a zero-feasible order $\preceq_u$ for $\widetilde{\mathcal{P}}_u$. This will imply that $\mathcal{T}_u = \{\widetilde{P}_i(s^*_i) \mid i \in I_u\}$ is a feasible template.

Now let us assume that we have a collection of disjoint sets $\mathcal{I} = \{I_u \mid u \in U\}$. This will be especially useful when considering the LinMinTemp problem. We will therefore need to identify the set that contains a given element i and so we define $u(i)$ to be such that $i \in I_{u(i)}$. Since we are trying to find zero-feasible solutions, we will need to check the inequality (5.1.1) repeatedly. The main calculation will be finding the sum of the d_ℓ's. In order to speed up this calculation we define the values

$$\textsc{Left-Sum}(u, i) = \begin{cases} 0, & i = \textsc{null} \\ \sum_{\ell \prec_u i} d_\ell, & \text{else} \end{cases} \quad \text{and} \quad \textsc{Right-Sum}(u, i) = \begin{cases} 0, & i = \textsc{null} \\ \sum_{\ell \succ_u i} d_\ell, & \text{else.} \end{cases}$$

Now the summation from inequality (5.1.1) can be found using

$$\sum_{i \prec_u \ell \prec_u j} d_\ell = \textsc{Right-Sum}(u,i) - \textsc{Right-Sum}(u,j) - d_j = \textsc{Left-Sum}(u,j) - \textsc{Left-Sum}(u,i) - d_i.$$

Similarly, it will be useful to be able to walk the chain $(I_u, \preceq_u)$ and so we define the functions

$$\textsc{Right-Neighbor}(u,i) = \begin{cases} \min_{\preceq_u}\{\ell \in I_u\}, & i = \textsc{null} \\ \textsc{null}, & i = \max_{\preceq_u}\{\ell \in I_u\} \\ \min_{\preceq_u}\{\ell \in I_u \mid \ell \succ i\}, & \text{else} \end{cases}$$

and

$$\textsc{Left-Neighbor}(u,i) = \begin{cases} \max_{\preceq_u}\{\ell \in I_u\}, & i = \textsc{null} \\ \textsc{null}, & i = \min_{\preceq_u}\{\ell \in I_u\} \\ \max_{\preceq_u}\{\ell \in I_u \mid \ell \prec i\}, & \text{else.} \end{cases}$$

5.1.1 Kruskal Heuristic

Let us begin by considering the Linearized Minimum Templates Problem (LinMinTemp) defined in Section 3.1. Given a set of linear patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$, we wish to find a partition of I into a collection of disjoint sets $\mathcal{I} = \{I_u \mid u \in U\}$ such that $\mathcal{S}(\widetilde{P}_u) \neq \emptyset$ for each $u \in U$. To do this, we find a partition U and strict total orderings $\preceq_u$ which are zero-feasible for $\widetilde{\mathcal{P}}_u$ for all $u \in U$.

Recall that Kruskal's MST algorithm [1] works by joining together trees that span disjoint subsets of the given node set. In the same way, our heuristic will join together disjoint subsets I_u and I_v, and combine their respective zero-feasible orderings $\preceq_u$ and $\preceq_v$ so that the augmented ordering $\preceq_{uv}$ is a strict total order on $I_u \cup I_v$ and is zero-feasible for $\widetilde{\mathcal{P}}_{uv} = \{\widetilde{P}_i \mid i \in I_u \cup I_v\}$. Recall that, since $\preceq_u$ is a strict total order on I_u, $(I_u, \preceq_u)$ is a chain as is $(I_v, \preceq_v)$. We will join sets by linking two zero-feasible chains end-to-end. We achieve this by placing (say) set I_u before set I_v to create set I_{uv}; accordingly we define the augmented order $\preceq_{uv}$ via

$$i \prec_u j \Rightarrow i \prec_{uv} j, \qquad i \prec_v j \Rightarrow i \prec_{uv} j, \qquad i \prec_{uv} j, \quad \forall i \in I_u, j \in I_v. \tag{5.1.2}$$

That is, $\preceq_{uv}$ maintains the orderings $\preceq_u$ and $\preceq_v$ and requires all elements of I_u to precede all elements of I_v. We need to determine what conditions are necessary so that this new combined chain is zero-feasible. Because the order of all indices in I_u is maintained in the new set I_{uv}, the inequalities of the form (5.1.1) for a pair $i \prec_u j$ are unmodified under the new ordering $\preceq_{uv}$. In other words, since no index from I_v has been placed between i and j, we have

$$B'_{ij} \geq \sum_{i \prec_u \ell \prec_u j} d_\ell = \sum_{i \prec_{uv} \ell \prec_{uv} j} d_\ell.$$

The same holds for pairs from I_v. This means that we only need to consider $i \in I_u$ and $j \in I_v$ in checking (5.1.1) or equivalently (3.2.4).

To clarify this, consider the following example.

Example 5.1.1: Consider the set of linear patterns $\widetilde{\mathcal{P}}^6 = \{\widetilde{P}_i \mid i \in I\}$ where

$$\begin{aligned} I &= \{1,2,3,4,5\} & \widetilde{P}_1 &= (420,4,2) \\ \widetilde{P}_2 &= (420,15,2) & \widetilde{P}_3 &= (420,7,2) \\ \widetilde{P}_4 &= (420,10,2) & \widetilde{P}_5 &= (420,6,2). \end{aligned}$$

Let $I_1 = \{1,2,3\}$ and $I_2 = \{4,5\}$. Define $\preceq_1$ and $\preceq_2$ such that $1 \prec_1 2 \prec_1 3$ and $4 \prec_2 5$. The values B_{ij} can be represented in matrix form as

$$\left[B_{ij}\right] = \begin{bmatrix} B_{12} & B_{13} & B_{14} & B_{15} \\ & B_{23} & B_{24} & B_{25} \\ & & B_{34} & B_{35} \\ & & & B_{45} \end{bmatrix} = \begin{bmatrix} \frac{420}{60} & \frac{420}{28} & \frac{420}{20} & \frac{420}{12} \\ & \frac{420}{105} & \frac{420}{30} & \frac{420}{30} \\ & & \frac{420}{70} & \frac{420}{42} \\ & & & \frac{420}{30} \end{bmatrix} = \begin{bmatrix} 7 & 15 & 21 & 35 \\ & 4 & 14 & 14 \\ & & 6 & 10 \\ & & & 14 \end{bmatrix}.$$

Note that

$$B_{12} = 7 \geq 4 = d_1 + d_2$$

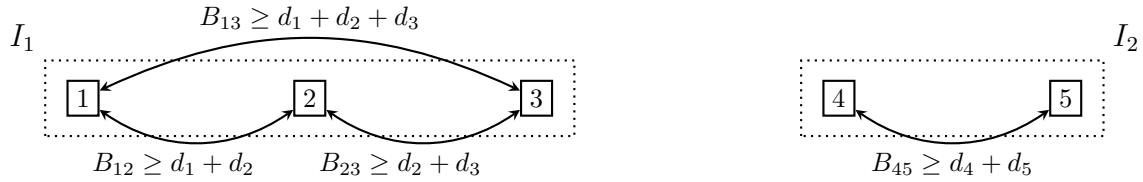
$$B_{23} = 4 \geq 4 = d_2 + d_3$$

$$B_{13} = 15 \geq 6 = d_1 + d_2 + d_3$$

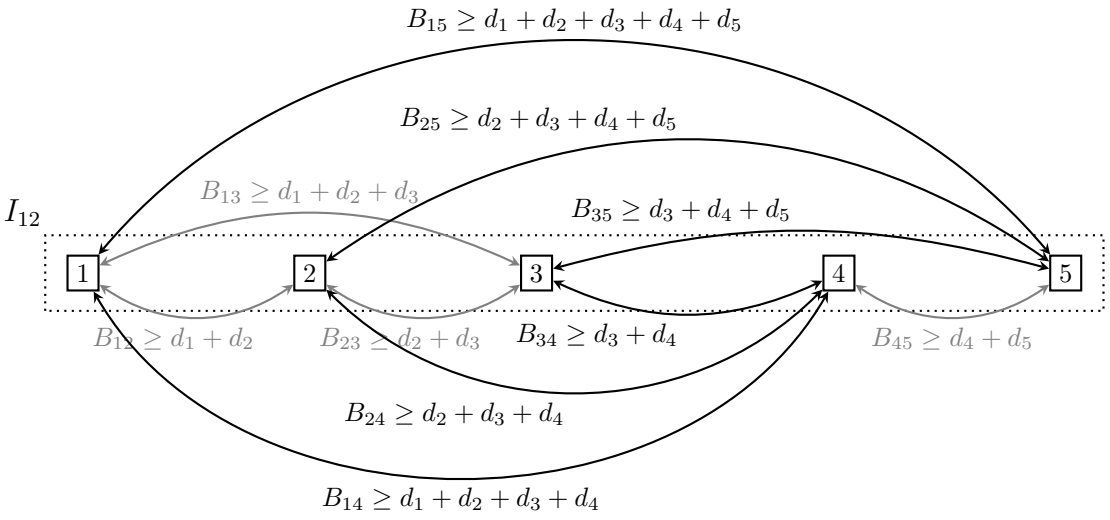
so $\preceq_1$ is zero-feasible for $\widetilde{\mathcal{P}}_1^6$. Additionally,

$$B_{45} = 14 \geq 4 = d_4 + d_5$$

so $\preceq_2$ is zero-feasible for $\widetilde{\mathcal{P}}_2^6$. This situation is shown in Figure 5.1.1(a).



5.1.1(a): The zero-feasible chains I_1 and I_2



5.1.1(b): The merged zero-feasible chain I_{12}

Figure 5.1.1: $\widetilde{\mathcal{P}}^6$: Joining two zero-feasible chains

We now wish to join I_1 and I_2. This will involve creating the merged set

$$I_{12} = I_1 \cup I_2 = \{1, 2, 3, 4, 5\}$$

and defining the ordering $\preceq_{12}$ where $1 \preceq_{12} 2 \preceq_{12} 3 \preceq_{12} 4 \preceq_{12} 5$. Figure 5.1.1(b) shows all inequalities of the form (3.2.4) that need to hold in order for $\preceq_{12}$ to be zero-feasible for $\widetilde{\mathcal{P}}^6$. Notice that four of these inequalities are duplicated exactly from $\preceq_1$ and $\preceq_2$ and are greyed out in Figure 5.1.1(b). The only inequalities left to check are those where $i \in I_1$ and $j \in I_2$:

$$B_{14} = 21 \geq 8 = d_1 + d_2 + d_3 + d_4 \qquad B_{15} = 35 \geq 10 = d_1 + d_2 + d_3 + d_4 + d_5$$

$$B_{24} = 14 \geq 6 = d_2 + d_3 + d_4 \qquad B_{25} = 14 \geq 8 = d_2 + d_3 + d_4 + d_5$$

$$B_{34} = 6 \geq 4 = d_3 + d_4 \qquad B_{35} = 10 \geq 6 = d_3 + d_4 + d_5$$

Thus, the new chain $(I_{12}, \preceq_{12})$ is zero-feasible. The starting positions s_i^* are therefore valid for $\widetilde{\mathcal{P}}^6$. We calculate these as

$$s_1^* = \sum_{\ell \prec_{12} 1} d_\ell = 0 \qquad s_2^* = \sum_{\ell \prec_{12} 2} d_\ell = d_1 = 2$$

$$s_3^* = \sum_{\ell \prec_{12} 3} d_\ell = d_1 + d_2 = 4 \qquad s_4^* = \sum_{\ell \prec_{12} 4} d_\ell = d_1 + d_2 + d_3 = 6$$

$$s_5^* = \sum_{\ell \prec_{12} 5} d_\ell = d_1 + d_2 + d_3 + d_4 = 8$$

The resulting feasible template is shown in Figure 5.1.2.

Figure 5.1.2: $\widetilde{\mathcal{P}}^6$: The feasible template $\{\widetilde{P}_1(0), \widetilde{P}_2(2), \widetilde{P}_3(4), \widetilde{P}_4(6), \widetilde{P}_5(8)\}$

We now proceed to the details of the Kruskal heuristic which deals extensively with sets I_u. Each set will need to have a listing of the elements in that set and the ordering of the elements in that set. We initialize the heuristic by placing each index $i \in I$ in its own subset. As a result, we will need a procedure MAKE-SET(p) that forms the initial set $I_u = \{p\}$ with a null ordering. We will also

need a procedure FIND-SET(p) that outputs the set $I_{u(p)}$ containing p. We then wish to join sets together and so we require procedure JOIN-SET(p, q) that finds the set I_u containing p and the set I_v containing q and merges these sets by placing I_u before I_v in the new order $\preceq_{uv}$. It does this by updating U so that FIND-SET(p) = FIND-SET$(q) = I_u \cup I_v$, and by creating $\preceq_{uv}$ as described in (5.1.2).

Recall that we only need to consider the new inequalities generated by joining two chains: those corresponding to the pair (i, j) where $i \in I_u$ and $j \in I_v$. Note that the summation in these inequalities is of the form $\sum_{i \prec_{uv} \ell \prec_{uv} j} d_\ell$. However, those indices between i and j with respect to $\preceq_{uv}$ are exactly those indices in I_u that are to the right of i and those indices in I_v to the left of j. Thus, we can calculate this sum before potentially merging I_u and I_v as

$$\sum_{i \prec_{uv} \ell \prec_{uv} j} d_\ell = \text{RIGHT-SUM}(u, i) + \text{LEFT-SUM}(v, j).$$

In this way, we can test whether two zero-feasible chains can merge to become a larger zero-feasible chain. To do this test, we run routine KRUSKAL-CHECK-ZERO-FEASIBILITY(p, q) shown in Algorithm 1[1], which returns true if $\left[(I_{u(p)}, \preceq_{u(p)}), (I_{u(q)}, \preceq_{u(q)})\right]$ would be a zero-feasible chain and false otherwise.

Algorithm 1 KRUSKAL-CHECK-ZERO-FEASIBILITY(p, q)

```
    I_u ← FIND-SET(p)
    I_v ← FIND-SET(q)
 3: for all i ∈ I_u do
      for all j ∈ I_v do
        if B'_ij < RIGHT-SUM(u, i) + LEFT-SUM(v, j) then
 6:       return FALSE
        end if
      end for
 9: end for
    return TRUE
```

One more detail that needs to be pointed out is that Theorem 3.1.19 guarantees that, if $\preceq$ is zero-feasible for a set of patterns $\widetilde{\mathcal{P}}$, then its inverse order $\succeq$ is also zero-feasible for $\widetilde{\mathcal{P}}$. Thus when joining two zero-feasible chains $(I_u, \preceq_u)$ and $(I_v, \preceq_v)$, there are four ways we can join them end-to-end to

[1]Note that all algorithms presented here have every third line numbered. This will be useful later when we discuss changes that can be made to the algorithms for alternative formulations.

possibly create a larger zero-feasible chain, namely

$$\big[(I_u, \preceq_u), (I_v, \preceq_v)\big], \qquad \big[(I_u, \preceq_u), (I_v, \succeq_v)\big],$$
$$\big[(I_v, \preceq_v), (I_u, \preceq_u)\big], \qquad \big[(I_v, \succeq_v), (I_u, \preceq_u)\big].$$

To implement this in our Kruskal heuristic, we employ a routine REVERSE(i) that reverses the ordering of a given chain $I_{u(i)}$.

Algorithm 2 describes the Kruskal heuristic for constructing a partition of I into zero-feasible chains. It requires a data structure H that stores pairs of indices from I. H needs to support two operations: INSERT(H, i, j) that adds pair (i, j) to H, and EXTRACT-MIN(H) that removes and returns a pair $(p, q) \in H$ such that

$$B'_{pq} \leq B'_{ij}, \quad \forall (i, j) \in H.$$

Algorithm 2 KRUSKAL-CONSTRUCTION($\widetilde{\mathcal{P}}$)

```
    for all i ∈ I do
      MAKE-SET(i)
3:  end for
    H ← ∅
    for all i ∈ I do
6:    for all j ∈ I \ {i} do
        INSERT(H, i, j)
      end for
9:  end for
    while H ≠ ∅ do
      (p, q) ← EXTRACT-MIN(H)
12:   for ℓ = 1..2 do
        if FIND-SET(p) ≠ FIND-SET(q) then
          if KRUSKAL-CHECK-ZERO-FEASIBILITY(p, q) = TRUE then
15:         JOIN-SET(p, q)
          else if KRUSKAL-CHECK-ZERO-FEASIBILITY(q, p) = TRUE then
            JOIN-SET(q, p)
18:       else
            REVERSE(q)
          end if
21:     end if
      end for
    end while
24: return (I_u, ⪯_u),   ∀u ∈ U
```

The Kruskal heuristic first creates a set $I_u = \{i\}$ for each element $i \in I$, and then it populates the data structure H with all possible pairs $i \neq j \in I$. We then extract all pairs (p, q) from H in ascending order of B'_{pq} and, if p and q are not already in the same chain, we try to join chains $I_{u(p)}$ and $I_{u(q)}$. Finally, once all pairs have been tried, we output the sets I_u and their orderings $\preceq_u$.

Theorem 5.1.2 *Given $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$, the sets I_u and orders $\preceq_u$ returned by the heuristic* KRUSKAL-CONSTRUCTION($\widetilde{\mathcal{P}}$) *are such that $\preceq_u$ is zero-feasible for $\widetilde{\mathcal{P}}_u = \{\widetilde{P}_i \mid i \in I_u\}$ for all $u \in U$.*

Proof: Vacuously, the null ordering is zero-feasible for the initial set $I_u = \{i\}$ for all $i \in I$. Then, since each call of the operation JOIN-SET(p, q) is preceded by a successful call to KRUSKAL-CHECK-ZERO-FEASIBILITY(p, q), we know that each merged set created by JOIN-SET(p, q) was also zero-feasible. Thus, all chains returned by KRUSKAL-CONSTRUCTION($\widetilde{\mathcal{P}}$) must be zero-feasible. ■

Notice that the Kruskal heuristic could be used as a heuristic for the Linearized Feasible Fit problem as well. If the KRUSKAL-CONSTRUCTION($\widetilde{\mathcal{P}}$) routine terminates with $|U| = 1$ or $\mathcal{I} = \{I\}$ then all patterns can be fit on the same template and the s_i^*'s are valid starting positions. Additionally, by taking the largest set I_u, the KRUSKAL-CONSTRUCTION($\widetilde{\mathcal{P}}$) routine could be used as a heuristic for the Linearized Maximum Template problem.

5.1.2 Prim Heuristic

We now turn our attention to the Linearized Maximum Template Problem. Recall that, given a set of linearized patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$, this involves finding a largest subset $I_1 \subseteq I$ such that there exist valid starting positions for $\widetilde{\mathcal{P}}_1 = \{\widetilde{P}_i \mid i \in I_1\}$. We take an incremental construction approach here, by adding *one* pattern at a time to an existing zero-feasible chain. As before, we would like to keep pairs of patterns with small B'_{ij} close together in the ordering. Again, we turn to minimum spanning tree algorithms for inspiration, this time to the classical algorithm of Jarník and Prim [1].

In this heuristic we keep track of the current subset I_1 and the associated current zero-feasible ordering $\preceq_1$. Repeatedly, we will attempt to add index $t \notin I_1$ to I_1 between two successive patterns p and q in I_1.[2] Care will need to be taken when adding t to I_1 to keep $\preceq_1$ both a strict total order on

[2]Note that we can also add to the ends of I_1 by setting either p or q equal to NULL.

the expanded I_1 and a zero-feasible ordering for the new $\widetilde{\mathcal{P}}_1$. Recall that, in the Kruskal heuristic, we only needed to check the new inequalities of the form (3.2.4) generated by merging two sets. In the Prim heuristic we still need to check that these new inequalities hold. That is, if t is being inserted between p and q, we need to ensure that

$$B'_{it} \geq \sum_{i \prec_1 \ell \preceq_1 p} d_\ell = \textsc{Right-Sum}(1, i) - \textsc{Right-Sum}(1, p), \quad \forall i \preceq_1 p$$

and

$$B'_{tj} \geq \sum_{q \preceq_1 \ell \prec_1 j} d_\ell = \textsc{Left-Sum}(1, j) - \textsc{Left-Sum}(1, q), \quad \forall j \succeq_1 q.$$

However, the difference in the Prim heuristic is that we may also affect some of the existing inequalities by inserting t into I_1. That is, we already know that

$$B_{ij} \geq \sum_{i \preceq_1 \ell \preceq_1 j} d_\ell, \quad \forall i \prec_1 j$$

since $\preceq_1$ is zero-feasible but, to ensure that $(I_1, \preceq_1)$ will remain zero-feasible after inserting t between p and q, we need to check that

$$B_{ij} \geq d_t + \sum_{i \preceq_1 \ell \preceq_1 j} d_\ell, \quad \forall i \preceq_1 p, \;\; j \succeq_1 q.$$

In the interest of verifying that the above inequalities hold after the insertion of t, define

$$\textsc{Slack}(i, j) = \begin{cases} B_{ij} - \displaystyle\sum_{i \preceq_1 \ell \preceq_1 j} d_\ell, & i \prec j \\ \infty, & \text{else.} \end{cases}$$

Thus, when inserting t between p and q, if $\textsc{Slack}(i, j) \geq d_t$ for all $i \preceq_1 p$ and $j \succeq_1 q$, then none of the existing inequalities (3.2.4) will be violated. Thus, if

$$d_t \leq \min\Big\{ \textsc{Slack}(i, j) \;\Big|\; i \preceq_1 p, \; j \succeq_1 q \Big\} \equiv \textsc{Min-Slack}(p, q) \tag{5.1.3}$$

then no existing inequality will be violated.

To clarify, consider the following example.

Example 5.1.3: Recall the set of linear patterns from Example 5.1.1, $\widetilde{\mathcal{P}}^6 = \{\widetilde{P}_i \mid i \in I\}$ where

$$I = \{1, 2, 3, 4, 5\} \qquad \widetilde{P}_1 = (420, 4, 2)$$
$$\widetilde{P}_2 = (420, 15, 2) \qquad \widetilde{P}_3 = (420, 7, 2)$$
$$\widetilde{P}_4 = (420, 10, 2) \qquad \widetilde{P}_5 = (420, 6, 2).$$

Let $I_1 = \{1, 2, 3, 5\}$ and let $\preceq_1$ be such that $1 \prec_1 2 \prec_1 3 \prec_1 5$. The values for B_{ij} are

$$\left[B_{ij}\right] = \begin{bmatrix} & B_{12} & B_{13} & B_{14} & B_{15} \\ & & B_{23} & B_{24} & B_{25} \\ & & & B_{34} & B_{35} \\ & & & & B_{45} \\ & & & & \end{bmatrix} = \begin{bmatrix} & 7 & 15 & 21 & 35 \\ & & 4 & 14 & 14 \\ & & & 6 & 10 \\ & & & & 14 \\ & & & & \end{bmatrix}.$$

Note that

$$B_{12} = 7 \geq 4 = d_1 + d_2 \qquad B_{13} = 15 \geq 6 = d_1 + d_2 + d_3$$
$$B_{23} = 4 \geq 4 = d_2 + d_3 \qquad B_{25} = 14 \geq 6 = d_2 + d_3 + d_5$$
$$B_{35} = 10 \geq 4 = d_3 + d_5 \qquad B_{15} = 35 \geq 8 = d_1 + d_2 + d_3 + d_5.$$

Thus $(I_1, \preceq_1)$ is a zero-feasible chain. This situation is shown in Figure 5.1.3(a). We then wish to insert 4 between 3 and 5. This forms $I_2 = \{1, 2, 3, 4, 5\}$ and $\preceq_2$ such that $1 \prec_2 2 \prec_2 3 \prec_2 4 \prec_2 5$. To ensure that $(I_2, \preceq_2)$ is a zero-feasible chain we first check to see whether the *existing* inequalities would be violated by inserting 4. The inequalities that are affected are shown in red in Figure 5.1.3(b). The greyed out inequalities are not affected by inserting 4 and so their feasibility is implied by the zero-feasibility of $\preceq_1$. We calculate $\textsc{Slack}(i, j)$ for all appropriate pairs i, j:

$$\textsc{Slack}(1, 5) = 35 - (2 + 2 + 2 + 2) = 27$$
$$\textsc{Slack}(2, 5) = 14 - (2 + 2 + 2) = 8$$
$$\textsc{Slack}(3, 5) = 10 - (2 + 2) = 6.$$

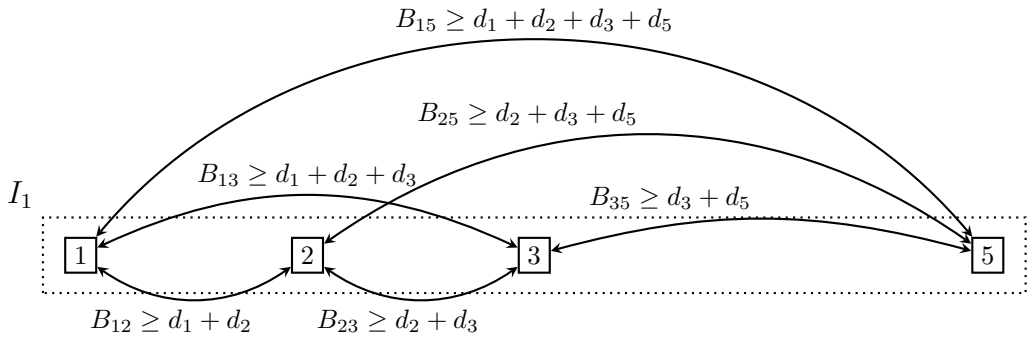
Thus, since

$$\textsc{Min-Slack}(3,5) = \min\{27, 8, 6\} = 6 \geq 2 = d_4$$

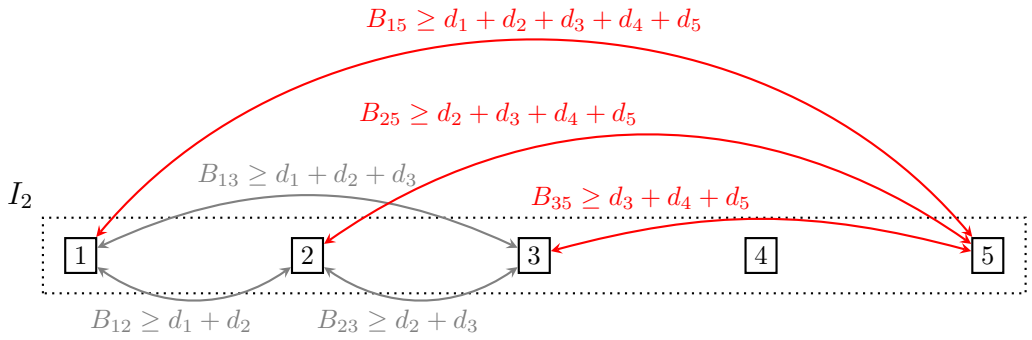
none of the existing inequalities will be violated by inserting 4 into I_1. Next we check the *new* inequalities created by inserting 4 into I_1. These new inequalities are shown in blue in Figure 5.1.3(c). Thus, since

$$B_{14} = 21 \geq 8 = d_1 + d_2 + d_3 + d_4 \qquad B_{24} = 14 \geq 6 = d_2 + d_3 + d_4$$
$$B_{34} = 6 \geq 4 = d_3 + d_4 \qquad B_{45} = 14 \geq 4 = d_4 + d_5,$$

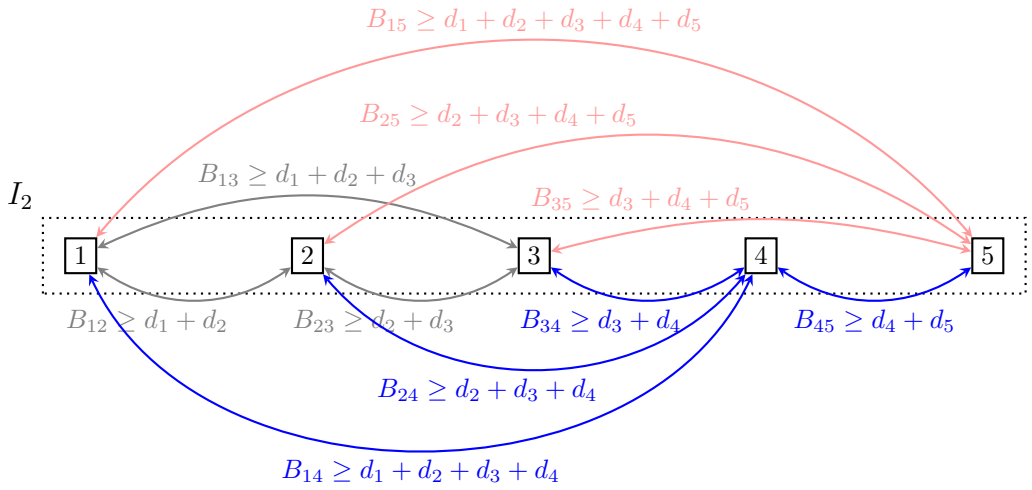
$(I_2, \preceq_2)$ is a zero-feasible chain. As a result, the s_i^*'s calculated in Example 5.1.1 are again valid for $\widetilde{\mathcal{P}}^6$ and the resulting feasible template is the same as that shown in Figure 5.1.2.



5.1.3(a): The zero-feasible chain I_1



5.1.3(b): Changes to the existing inequalities



5.1.3(c): The new inequalities

Figure 5.1.3: $\widetilde{\mathcal{P}}^6$: Inserting into a zero-feasible chain

To handle the details of inserting the new index t into I_u we developed a routine ADD-PATTERN(t, p, q) to modify the set I_u. The arguments are t, the index to be added to I_1, and p and q, the indices we wish to insert t between. The routine adds t to the set I_1 and updates $\preceq_1$ so that $i \prec_1 t$ for all $i \preceq_1 p$ and $j \succ_1 t$ for all $j \succeq_1 q$. Next we use routine PRIM-CHECK-ZERO-FEASIBILITY(t, p, q) shown in Algorithm 3 to check all appropriate inequalities described above to ensure the zero-feasibility of the augmented $\preceq_u$. This routine will also check the existing inequalities. Recall from (5.1.3) that this can be done more efficiently by using MIN-SLACK(p, q) which we calculate as

$$\text{MIN-SLACK}(p, q) = \min\Big\{\text{SLACK}(i, j) \;\Big|\; i \preceq_1 p, \;\; j \succeq_1 q\Big\}.$$

Algorithm 3 PRIM-CHECK-ZERO-FEASIBILITY(t, p, q)

```
    if d_t > MIN-SLACK(p, q) then
      return FALSE
 3: end if
    i ← p
    while i ≠ NULL do
 6:   if B'_it < RIGHT-SUM(1, i) − RIGHT-SUM(1, p) then
        return FALSE
      end if
 9:   i ← LEFT-NEIGHBOR(1, i)
    end while
    j ← q
12: while j ≠ NULL do
      if B'_tj < LEFT-SUM(1, j) − LEFT-SUM(1, q) then
        return FALSE
15:   end if
      j ← RIGHT-NEIGHBOR(1, j)
    end while
18: return TRUE
```

Since we may need to check all possible positions to successfully insert index t, we developed the routine PRIM-CHECK-ALL-FEASIBILITY(t) shown in Algorithm 4. This routine simply runs the subroutine PRIM-CHECK-ZERO-FEASIBILITY(t, p, q) for all possible adjacent pairs (p, q) in I_1 (including the two end positions). It returns valid positions $(\overline{p}, \overline{q})$ such that PRIM-CHECK-ZERO-FEASIBILITY$(t, \overline{p}, \overline{q})$ yields TRUE, or (NULL, NULL) if no such positions exist.

Algorithm 4 PRIM-CHECK-ALL-FEASIBILITY(t)

```
    p ← NULL
    q ← RIGHT-NEIGHBOR(1, p)
 3: while q ≠ NULL do
      if PRIM-CHECK-ZERO-FEASIBILITY(t, p, q) = TRUE then
        return (p, q)
 6:   end if
      p ← q
      q ← RIGHT-NEIGHBOR(1, p)
 9: end while
    if PRIM-CHECK-ZERO-FEASIBILITY(t, p, q) = TRUE then
      return (p, q)
12: end if
    return (NULL, NULL)
```

Algorithm 5 PRIM-CONSTRUCTION$(\widetilde{\mathcal{P}}, r)$

```
    I_1 ← {r}
    H ← ∅
 3: for all j ∈ I \ {r} do
      INSERT(H, j, B'_rj)
    end for
 6: ⪯_1 ← ∅
    while H ≠ ∅ do
      t ← EXTRACT-MIN(H)
 9:   (p, q) ← PRIM-CHECK-ALL-FEASIBILITY(t)
      if p ≠ NULL and q ≠ NULL then
        ADD-PATTERN(t, p, q)
12:     for all j ∈ H do
          if B'_tj < KEY(j) then
            UPDATE-KEY(H, j, B'_tj)
15:       end if
        end for
      end if
18: end while
    return (I_1, ⪯_1)
```

Algorithm 5 describes the Prim heuristic for constructing a maximal zero-feasible chain I_1. We initialize I_1 by inserting a starting index $r \in I$. There are a number of different ways to choose r, which are discussed in Section 5.1.3.1. The heuristic then inserts all other elements of I into H. In the main loop of the heuristic we remove an index t from H such that

$$t = \operatorname*{argmin}_{j \in H} \Big\{ \min_{i \in I_1} \{B'_{ij}\} \Big\}.$$

We then try to insert t into every possible position in I_1. If valid positions (p, q) are returned by PRIM-CHECK-ALL-FEASIBILITY(t), then the heuristic inserts index t into the appropriate position of I_1. If no feasible positions are found then the current index is discarded. Finally, the heuristic terminates when all indices in H have been tried and it outputs I_1 and $\preceq_1$.

We treat the set H as a priority queue, sorted by the value KEY$(j) = \min\{B'_{ij} \mid i \in I_1\}$. Thus H will need to support INSERT(H, j, α) which inserts j into H and sets KEY$(j) = \alpha$. This value will initially be B'_{rj} where r is the initial index in I_1. It will also need to support EXTRACT-MIN(H) which removes and returns the element t in H with minimum KEY(j) value. Additionally, in order for H to be consistent after we change the set I_1, we need to update the value of KEY(j) for all $j \in H$. As a result, H must support UPDATE-KEY(H, j, α) which sets the value of KEY(j) to α and re-sorts the queue so that the new (minimum) element can be returned.

Theorem 5.1.4 *Given $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ and $r \in I$, the chain $(I_1, \preceq_1)$ returned by* PRIM-CONSTRUCTION$(\widetilde{\mathcal{P}}, r)$ *is zero-feasible.*

Proof: Vacuously, the null ordering is zero-feasible for the initial set $I_1 = \{r\}$. Then before each call to ADD-PATTERN(t, p, q) we make a call to PRIM-CHECK-ALL-FEASIBILITY(t) that returns (p, q). Therefore, since that routine must have made a call to PRIM-CHECK-ZERO-FEASIBILITY(t, p, q) that returned TRUE, we know that the chain created by ADD-PATTERN(t, p, q) is zero-feasible. Thus inductively, the chain returned by PRIM-CONSTRUCTION$(\widetilde{\mathcal{P}}, r)$ must be zero-feasible. ■

Notice that the Prim construction procedure can be used as a heuristic for the Linearized Feasible Fit problem as well. If the PRIM-CONSTRUCTION$(\widetilde{\mathcal{P}}, r)$ routine terminates with $|I_1| = |I|$ then all patterns can be fit on the same template and the associated s_i^*'s are valid starting positions. Additionally, by finding the largest subset I_1 and then iteratively running the heuristic on the remaining indices $I \setminus I_1$, PRIM-CONSTRUCTION$(\widetilde{\mathcal{P}}, r)$ can be used as a heuristic for the Linearized Minimum Templates problem.

5.1.3 Implementation

To implement the Kruskal heuristic we employ a disjoint-set data structure. If this is implemented as a disjoint-set forest with path compression and union-by-rank [9], then the set operations MAKE-SET(p), FIND-SET(p), and JOIN-SET(p, q) each run in $O(\alpha(|I|))$ amortized time where α is the Inverse Ackermann function. There is additional bookkeeping needed when JOIN-SET(p, q) is run, such as updating LEFT-SUM(u, i) and RIGHT-SUM(u, i). However, we know that after JOIN-SET(p, q) is run, we will access the value LEFT-SUM(u, i) again before the next call to JOIN-SET(p, q). Thus, we can do a 'lazy' update, storing incremental values Δ_{left} and Δ_{right} for each set I_u, so that when the next call is made to RIGHT-SUM(u, i), the Δ values can be added to the LEFT- and RIGHT-SUM values. Thus we amortize this bookkeeping effort into the RIGHT-SUM(u, i) and LEFT-SUM(u, i) calls, keeping the amortized running time of JOIN-SET(p, q) to $O\big(\alpha(|I|)\big)$. The total running time for all set operations is then dominated by the $O(|I|^2)$ calls to FIND-SET(p) giving us $O\big(|I|^2\alpha(|I|)\big)$ time overall.

The total number of calls to RIGHT-SUM(u, i) and LEFT-SUM(u, i) can also be reduced. It is possible that the priority queue H returns a pair (p, q) where p and q are already in the same chain, or in chains that have already been compared and found incompatible for merging. Thus, if we keep a boolean matrix of all (p, q) pairs in I initialized to FALSE, and set the entry to TRUE once p and q have been compared in the routine KRUSKAL-CHECK-ZERO-FEASIBILITY(p, q), then the pair (p, q) can only be checked a maximum of 4 times. This reduces the total amount of work done by these calls to $O(|I|^2)$.

The operations on the priority queue H also need to be accounted for. If implemented as a binary heap, then each operation runs in $O(\log |I|)$ time. However, since we have all the elements for the heap at the start, we can actually construct the heap in $O(|I|^2)$ time. Thus, the total time taken for the heap operations is dominated by the calls to EXTRACT-MIN(H) which take a total of $O(|I|^2 \log |I|)$ time. The last operation that needs to be considered is the REVERSE(q) routine. If we practice a 'lazy' reverse then we only need to actually reverse the elements when merging one chain that is reversed and one that is not. In this case we can always reverse the smaller chain before merging which gives a total of $O(|I|)$ time. Thus the total running time for the KRUSKAL-

CONSTRUCTION($\widetilde{\mathcal{P}}$) heuristic is

$$O\big(|I|^2\alpha(|I|)\big) + O(|I|^2) + O(|I|^2 \log |I|) + O(|I|) = O(|I|^2 \log |I|).$$

To implement the Prim heuristic we cannot rely on the same analysis for updating the LEFT-SUM$(1, i)$ and RIGHT-SUM$(1, i)$ values. This needs to be done in the ADD-PATTERN(t, p, q) routine and thus takes $O(|I_1|^2)$ time for each call to ADD-PATTERN(t, p, q). This time is dominated by the effort needed to update the MIN-SLACK(p, q) parameter. While this does save time in the PRIM-CHECK-ZERO-FEASIBILITY(t, p, q) routine, it takes a total of $O(|I_1|^2)$ effort to keep it updated for each call to ADD-PATTERN(t, p, q). The PRIM-CHECK-ZERO-FEASIBILITY(t, p, q) routine itself then takes $O(|I_1|)$ time and is called $O(|I_1|)$ times by the PRIM-CHECK-ALL-FEASIBILITY(t) routine.

The priority queue H can again be implemented as a binary heap. This means that it takes $O(|I|)$ time to initialize and $O(\log |I|)$ time for each EXTRACT-MIN(H) call. An iteration of the PRIM-CONSTRUCTION$(\widetilde{\mathcal{P}}, \preceq)$ routine then requires a single call to EXTRACT-MIN(H), a call to the PRIM-CHECK-ALL-FEASIBILITY(t) routine, a possible call to ADD-PATTERN(t, p, q) and $|I \setminus I_1|$ calls to UPDATE-KEY(H, i, α). Thus the total running time of the heuristic is

$$\begin{aligned} O(|I|) + \sum_{\ell=1}^{|I|} \Big[O(\log |I|) + O(\ell^2) + O(\ell^2) + (|I| - \ell)O(\log |I|)\Big] &= O(|I|) + O(|I| \log |I|) + \sum_{\ell=1}^{|I|} \big(O(\ell^2)\big) \\ &= O(|I|) + O(|I| \log |I|) + O(|I|^3) \\ &= O(|I|^3). \end{aligned}$$

5.1.3.1 Initializing the Prim Heuristic

Whereas the Kruskal heuristic starts with each index in its own set, the Prim heuristic needs to be initialized by placing one index r in the set I_1. Prim's MST algorithm can be initialized with any node in the graph, and it will find a minimum spanning tree. However, since our Prim heuristic builds a chain and not a tree, starting it in different places may lead to different solutions. As a result, the decision about where to start the Prim heuristic is an important one. One strategy is to start with a random index; however other options are more successful in practice. Another idea is to start with one of the patterns in the pair with the smallest B'_{ij} value. Which one is inconsequential

as the heuristic will add the other member of the pair next. Alternatively, since the heuristic has polynomial running time, we can simply run the algorithm $|I|$ times, starting with each index in turn, and choose the best solution appearing.

5.1.3.2 Modifications for Contiguous Ordering

Although both the Kruskal and Prim heuristics described in this section are based on finding zero-feasible orderings, they can be adapted very easily to find contiguous orderings instead. The reason for this is that the inequalities (3.2.6) which determine a contiguous ordering are very similar to the inequalities (3.2.4) which define a zero-feasible ordering. Thus, the Kruskal heuristic can be modified to find contiguous orderings simply by changing line 5 of the routine KRUSKAL-CHECK-ZERO-FEASIBILITY(p, q). The updated routine is shown in Algorithm 6. The new routine KRUSKAL-CHECK-CONTIGUOUS-FEASIBILITY(p, q) can then be used in place of KRUSKAL-CHECK-ZERO-FEASIBILITY(p, q) in lines 14 and 16 of the routine KRUSKAL-CONSTRUCTION$(\widetilde{\mathcal{P}})$ to produce a partition of I into contiguous chains. Note that this does not change the overall running time of the heuristic.

Algorithm 6 KRUSKAL-CHECK-CONTIGUOUS-FEASIBILITY(p, q)

```
   I_u ← FIND-SET(p)
   I_v ← FIND-SET(q)
3: for all i ∈ I_u do
     for all j ∈ I_v do
       if B'_ij < (RIGHT-SUM(u, i) + LEFT-SUM(v, j)) mod B_ij then
6:       return FALSE
       end if
     end for
9: end for
   return TRUE
```

In a similar way, the Prim heuristic can also be modified to find a maximal contiguous chain. Lines 6 and 13 of routine PRIM-CHECK-ZERO-FEASIBILITY(t, p, q) need the same modification as in KRUSKAL-CHECK-ZERO-FEASIBILITY(p, q) to check the feasibility of the new inequalities formed by inserting t into I_1. However, the major change is that we can no longer easily check the existing inequality simply using the minimum SLACK(i, j) when inserting t between p and q in I_1. Instead,

we need to ensure that

$$\big(\textsc{Slack}(i,j) - d_t\big) \bmod B_{ij} \leq B'_{ij}, \quad \forall i \preceq_1 p, \;\; j \succeq_1 q.$$

The corresponding routine Prim-Check-Contiguous-Feasibility(t,p,q) is shown in Algorithm 7. This routine can be used in place of Prim-Check-Zero-Feasibility(t,p,q) in lines 4 and 10 of routine Prim-Check-All-Feasibility(t).

It should be noted that updating a matrix of Slack(i,j) values takes the same amount of time as updating the Min-Slack(p,q) value but checking the existing inequalities now takes $O(|I_1|)$ time for each call to Prim-Check-Contiguous-Feasibility(t,p,q) instead of $O(1)$ time for each call to Prim-Check-Zero-Feasibility(t,p,q). However, this does not affect the overall asymptotic running time of the Prim heuristic.

Algorithm 7 Prim-Check-Contiguous-Feasibility(t,p,q)

```
    i ← p
    while i ≠ NULL do
3:    j ← q
      while j ≠ NULL do
        if B'_ij < (Slack(i,j) − d_t) mod B_ij then
6:        return FALSE
        end if
        j ← Right-Neighbor(1,j)
9:    end while
      i ← Left-Neighbor(1,i)
    end while
12: i ← p
    while i ≠ NULL do
      if B'_it < (Right-Sum(1,i) − Right-Sum(1,p)) mod B_ij then
15:     return FALSE
      end if
      i ← Left-Neighbor(1,i)
18: end while
    j ← q
    while j ≠ NULL do
21:   if B'_tj < (Left-Sum(1,j) − Left-Sum(1,q)) mod B_ij then
        return FALSE
      end if
24:   j ← Right-Neighbor(1,j)
    end while
    return TRUE
```

5.2 Cycle-Based Heuristics

The heuristics presented in Section 5.1 are specifically designed to find zero-feasible or contiguously ordered solutions. While they perform well at this task, the number of sets of patterns that do not have zero-feasible or contiguous solutions grows as the number of patterns in the set grows[3]. To address this limitation, we present another heuristic where, instead of the set K having a specific structure and trying to find a feasible order $\preceq$, we hold the ordering constant and try to find a feasible set K. To achieve this, we utilize the network formulation of the problem of finding valid starting positions for a given set of patterns. Recall from Section 3.1 that, if we can find a set K such that $G(\widetilde{\mathcal{P}}, \preceq, K)$ has no negative cycles, then we can find shortest paths from (or to) some given root node r in $G(\widetilde{\mathcal{P}}, \preceq, K)$, and the lengths of those shortest paths yield valid starting positions for the set $\widetilde{\mathcal{P}}$. Thus, given a set of patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ and a total order $\preceq$ on I, the goal of this heuristic is to find a maximal subset I_u of I and a set of integers K_u such that $G(\widetilde{\mathcal{P}}_u, \preceq, K_u)$ has no negative cycles. We do this by adding patterns to the end of subset I_u one at a time[4].

Assume that we have $I_1 \subseteq I$ and a set $K_1 = \{k^1_{ij} \in \mathbb{Z} \mid i \prec j \in I_1\}$ such that $G(\widetilde{\mathcal{P}}_1, \preceq, K_1)$ has no negative cycles. We now wish to add index t to I_1 forming $I_2 = I_1 \cup \{t\}$ and expand the total order $\preceq$ so that $i \prec t$ for all $i \in I_1$. We then need to find K_2 such that $G(\widetilde{\mathcal{P}}_2, \preceq, K_2)$ has no negative cycles. Initially, let $K_2 = \{k^2_{ij} \mid i \prec j \in I_2\}$ where

$$k^2_{ij} = \begin{cases} k^1_{ij}, & i \prec j \in I_1 \\ 0, & \text{else.} \end{cases}$$

Now note that $G(\widetilde{\mathcal{P}}_1, \preceq, K_1) \subset G(\widetilde{\mathcal{P}}_2, \preceq, K_2)$, and since $G(\widetilde{\mathcal{P}}_1, \preceq, K_1)$ has no negative cycle, any negative cycle in $G(\widetilde{\mathcal{P}}_2, \preceq, K_2)$ must pass through the new node t. We wish to eliminate any negative cycles in $G(\widetilde{\mathcal{P}}_2, \preceq, K_2)$ by modifying the k^2_{ij}'s. However, we also want to maintain the feasibility of $G(\widetilde{\mathcal{P}}_1, \preceq, K_1)$, so we will only modify those k^2_{ij}'s that involve t, i.e., k^2_{it}. We proceed as follows. Let C be a negative cycle in $G(\widetilde{\mathcal{P}}_2, \preceq, K_2)$. There must exist i, j in I_1 such that (i,t), $(t,j) \in C$. Since $i \prec t$ for all $i \in I_1$, the weights on these arcs are $w_{it} = (k^2_{it}+1)B_{it} - d_t$ and $w_{tj} = -k^2_{jt}B_{jt} - d_j$. The

[3]Chapter 6 discusses the performance of the Kruskal and Prim heuristics, as well as the density of zero-feasible and contiguous solutions.

[4]The order in which we add patterns can lead to different solutions. See Section 5.2.1.2 for a discussion of which orderings empirically lead to better performance.

values B_{it}, B_{jt}, d_t, and d_j are all positive and so, to make $w[C]$ non-negative, we need to increase k^2_{it} or decrease k^2_{jt}.

To clarify this process, consider the following example.

Example 5.2.1: Recall the set of linear patterns $\widetilde{\mathcal{P}}^6 = \{\widetilde{P}_i \mid i \in I\}$ from Example 5.1.1, where

$$I = \{1,2,3,4,5\} \qquad \widetilde{P}_1 = (420,4,2)$$
$$\widetilde{P}_2 = (420,15,2) \qquad \widetilde{P}_3 = (420,7,2)$$
$$\widetilde{P}_4 = (420,10,2) \qquad \widetilde{P}_5 = (420,6,2).$$

The values for B_{ij} are

$$\left[B_{ij}\right] = \begin{bmatrix} & B_{12} & B_{13} & B_{14} & B_{15} \\ & & B_{23} & B_{24} & B_{25} \\ & & & B_{34} & B_{35} \\ & & & & B_{45} \\ & & & & \end{bmatrix} = \begin{bmatrix} & 7 & 15 & 21 & 35 \\ & & 4 & 14 & 14 \\ & & & 6 & 10 \\ & & & & 14 \\ & & & & \end{bmatrix}.$$

Assume that $I_1 = \{2,3,4,1\}$ and let $\preceq$ be defined such that $2 \prec 3 \prec 4 \prec 1$ with

$$K_1 = \begin{bmatrix} & k^1_{23} & k^1_{24} & k^1_{21} \\ & & k^1_{34} & k^1_{31} \\ & & & k^1_{41} \\ & & & \end{bmatrix} = \begin{bmatrix} & 0 & 0 & 1 \\ & & 0 & 0 \\ & & & 0 \\ & & & \end{bmatrix}.$$

These parameters produce the feasible template $\mathcal{T}_1 = \{\widetilde{P}_2(0), \widetilde{P}_3(2), \widetilde{P}_4(4), \widetilde{P}_1(9)\}$, as shown in Figure 5.2.1(a). We wish to add pattern 5 to the current feasible template forming $I_2 = \{2,3,4,1,5\}$, with $2 \prec 3 \prec 4 \prec 1 \prec 5$. First, let us calculate the arc weights $W = (w_{ij})$ of the network $G(\mathcal{P}_2, \preceq, K_2)$

treating the k_{it}^2 as unknown:

$$W = \begin{bmatrix} & w_{23} & w_{24} & w_{21} & w_{25} \\ w_{32} & & w_{34} & w_{31} & w_{35} \\ w_{42} & w_{43} & & w_{41} & w_{45} \\ w_{12} & w_{13} & w_{14} & & w_{15} \\ w_{52} & w_{53} & w_{54} & w_{51} & \end{bmatrix}$$

$$= \begin{bmatrix} & & 2 & 12 & 12 & 14k_{25}^2 + 12 \\ -2 & & & 4 & 13 & 10k_{35}^2 + 8 \\ -2 & & -2 & & 19 & 14k_{45}^2 + 12 \\ -9 & & -2 & -2 & & 35k_{15}^2 + 33 \\ -14k_{25}^2 - 2 & & -10k_{35}^2 - 2 & -14k_{45}^2 - 2 & -35k_{15}^2 - 2 & \end{bmatrix}.$$

We then initialize K_2 as

$$K_2 = \begin{bmatrix} & k_{23}^2 & k_{24}^2 & k_{21}^2 & k_{25}^2 \\ & & k_{34}^2 & k_{31}^2 & k_{35}^2 \\ & & & k_{41}^2 & k_{45}^2 \\ & & & & k_{15}^2 \\ & & & & \end{bmatrix} = \begin{bmatrix} & 0 & 0 & 1 & 0 \\ & & 0 & 0 & 0 \\ & & & 0 & 0 \\ & & & & 0 \\ & & & & \end{bmatrix},$$

which gives arc weights

$$W' = \begin{bmatrix} & w'_{23} & w'_{24} & w'_{21} & w'_{25} \\ w'_{32} & & w'_{34} & w'_{31} & w'_{35} \\ w'_{42} & w'_{43} & & w'_{41} & w'_{45} \\ w'_{12} & w'_{13} & w'_{14} & & w'_{15} \\ w'_{52} & w'_{53} & w'_{54} & w'_{51} & \end{bmatrix} = \begin{bmatrix} & 2 & 12 & 12 & 12 \\ -2 & & 4 & 13 & 8 \\ -2 & -2 & & 19 & 12 \\ -9 & -2 & -2 & & 33 \\ -2 & -2 & -2 & -2 & \end{bmatrix}.$$

Relative to weights W', the cycle $C_1 = 1 \to 2 \to 3 \to 5 \to 1$ has weight

$$w'[C_1] = -9 + 2 + 8 - 2 = -1.$$

The two arcs on this cycle incident with 5 are $(3,5)$ and $(5,1)$. Thus, to correct this negative cycle, we can either increase k^2_{35} or decrease k^2_{15}. If we arbitrarily set $k^2_{35} = 1$, the arc weight matrix becomes

$$W'' = \begin{bmatrix} & w''_{23} & w''_{24} & w''_{21} & w''_{25} \\ w''_{32} & & w''_{34} & w''_{31} & w''_{35} \\ w''_{42} & w''_{43} & & w''_{41} & w''_{45} \\ w''_{12} & w''_{13} & w''_{14} & & w''_{15} \\ w''_{52} & w''_{53} & w''_{54} & w''_{51} & \end{bmatrix} = \begin{bmatrix} & 2 & 12 & 12 & 12 \\ -2 & & 4 & 13 & 18 \\ -2 & -2 & & 19 & 12 \\ -9 & -2 & -2 & & 33 \\ -2 & -12 & -2 & -2 & \end{bmatrix}.$$

Cycle C_1 now has weight

$$w''[C_1] = -9 + 2 + 18 - 2 = 9.$$

However, cycle $C_2 = 3 \to 2 \to 5 \to 3$ now has weight[5]

$$w''[C_2] = -2 + 12 - 12 = -2.$$

5.2.1(a): The feasible template $\{\widetilde{P}_2(0), \widetilde{P}_3(2), \widetilde{P}_4(4), \widetilde{P}_1(9)\}$

5.2.1(b): The feasible template $\{\widetilde{P}_2(0), \widetilde{P}_3(2), \widetilde{P}_4(4), \widetilde{P}_1(9), \widetilde{P}_5(16)\}$

Figure 5.2.1: $\widetilde{\mathcal{P}}^6$: Feasible templates, before and after the addition of pattern 5

The two arcs in this cycle incident with 5 are $(2,5)$ and $(5,3)$. To make $w[C_2]$ non-negative we can either increase k^2_{25} or decrease k^2_{35}. However, since we already increased k^2_{35}, decreasing it again would cause us to return to the previous situation. As a result we choose to increase k^2_{25} to 1. This

[5]Note that C_2 had weight $w'[C_2] = -2 + 12 - 2 = 8$ before we increased k^2_{35}. Thus, by eliminating one negative cycle we created another.

gives us new weights

$$W''' = \begin{bmatrix} & w'''_{23} & w'''_{24} & w'''_{21} & w'''_{25} \\ w'''_{32} & & w'''_{34} & w'''_{31} & w'''_{35} \\ w'''_{42} & w'''_{43} & & w'''_{41} & w'''_{45} \\ w'''_{12} & w'''_{13} & w'''_{14} & & w'''_{15} \\ w'''_{52} & w'''_{53} & w'''_{54} & w'''_{51} & \end{bmatrix} = \begin{bmatrix} & 2 & 12 & 12 & 26 \\ -2 & & 4 & 13 & 18 \\ -2 & -2 & & 19 & 12 \\ -9 & -2 & -2 & & 33 \\ -16 & -12 & -2 & -2 & \end{bmatrix}.$$

These weights however produce no negative cycles. Thus, with

$$K_2 = \begin{bmatrix} & k^2_{23} & k^2_{24} & k^2_{21} & k^2_{25} \\ & & k^2_{34} & k^2_{31} & k^2_{35} \\ & & & k^2_{41} & k^2_{45} \\ & & & & k^2_{15} \\ & & & & \end{bmatrix} = \begin{bmatrix} & 0 & 0 & 1 & 1 \\ & & 0 & 0 & 1 \\ & & & 0 & 0 \\ & & & & 0 \\ & & & & \end{bmatrix}$$

the resulting network $G(\mathcal{P}_2, \preceq, K_2)$ has no negative cycles. We can now find shortest paths in this network, say to node 2, giving us the feasible template shown in Figure 5.2.1(b)[6].

To carry out this cycle-based heuristic, we develop routine ELIMINATE-NEGATIVE-CYCLES(u, K_u, t) shown in Algorithm 8 which takes arguments u (the index of the current subset I_u of I), K_u (the current set of integers k^u_{ij}), and t (the index of the pattern to be added to the current feasible template). ELIMINATE-NEGATIVE-CYCLES(u, K_u, t) outputs K_v, a new set of integers k^v_{ij} such that $G\Big(\widetilde{\mathcal{P}}_v = \Big\{\widetilde{P}_i \mid i \in I_u \cup \{t\}\Big\}, \preceq, K_v\Big)$ has no negative cycles, or NULL if no such K_v could be found. In order to eliminate negative cycles we first need to be able to repeatedly find such cycles. For this we designed a routine FIND-NEGATIVE-CYCLE(v) that returns a cycle in $G(\widetilde{\mathcal{P}}_v, \preceq, K_v)$ of negative weight, or NULL if no such cycle exists[7]. To avoid cycling we add a flag DIRECTION(i) that can take values UP, DOWN, or NULL. Each time we attempt to add a new index t to the current template, these flags are set to the value NULL. We are then free to increase or decrease k^2_{it}. Once k^2_{it} has been either increased or decreased, we set DIRECTION(i) to UP or DOWN respectively. After

[6]Note that these patterns are the same as those in Example 5.1.1; however, here we develop a different solution that is not zero-feasible. Compare Figures 5.2.1(b) and 5.1.2.

[7]Several different options for such a routine are discussed in Section 5.2.1.1.

DIRECTION(i) = UP we never[8] decrease k^2_{it}, similarly when DIRECTION(i) = DOWN. To ensure that the heuristic terminates, we do not allow any k^2_{it} to exceed the bounds from Theorem 3.1.12.

Algorithm 8 ELIMINATE-NEGATIVE-CYCLES(u, K_u, t)

for all $i \in I_u$ **do**
  for all $j \succ i \in I_u$ **do**
    $k^v_{ij} \leftarrow k^u_{ij}$
  end for
  $k^v_{it} \leftarrow 0$
  DIRECTION$(i) \leftarrow$ NULL
end for
$f \leftarrow$ FALSE
while $f =$ FALSE **do**
  $f \leftarrow$ TRUE
  $C \leftarrow$ FIND-NEGATIVE-CYCLE(v)
  if $C =$ NULL **then**
    return K_v
  end if
  for $(i, j) \in C$ **do**
    if $j = t$ **and** DIRECTION$(i) \neq$ DOWN **and** $k^v_{it} < \dfrac{n_i}{(n_i, n_t)} - 1$ **then**
      $k^v_{it} \leftarrow k^v_{it} + 1$
      $f \leftarrow$ FALSE
    end if
    if $i = t$ **and** DIRECTION$(j) \neq$ UP **and** $k^v_{jt} > -\dfrac{n_t}{(n_j, n_t)}$ **and** $f =$ TRUE **then**
      $k^v_{jt} \leftarrow k^v_{jt} - 1$
      $f \leftarrow$ FALSE
    end if
  end for
end while
return NULL

Note that this routine tries to increase the total weight of a cycle by increasing (or decreasing) the value of k_{it} by one unit. This may not be sufficient to make the given cycle non-negative, so this cycle may have to be dealt with again. The routine can be easily modified so that k_{it} is increased (or decreased) by a sufficient amount to cause the cycle to become non-negative. However this requires calculating the total weight of the cycle and does not improve the worst-case running time of the heuristic. In this implementation, the choice was made to only increase k_{it} by one unit per iteration.

The overall routine CYCLE-CONSTRUCTION$(\widetilde{\mathcal{P}}, \preceq)$ is shown in Algorithm 9. The routine accepts as

[8]Note that, while k^2_{it} will never decrease again, a new entering index t' could be chosen later, DIRECTION(i) will be set to NULL, and $k^v_{it'}$ will be free to increase or decrease.

input a set of linear patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ and a total ordering $\preceq$ on I. It then selects r, the first index in I with respect to $\preceq$, and adds it to I_1. The next index $t \in I$, in the order given by $\preceq$, is selected and the algorithm tries to add t to the end of set I_1. If the routine ELIMINATE-NEGATIVE-CYCLES$(1, K_1, t)$ succeeds in finding a set K_2 such that $G(\widetilde{\mathcal{P}}_2, \preceq, K_2)$ has no negative cycles, then t is added to I_1, $\preceq_1$ is updated and the set K_1 is replaced with K_2. If ELIMINATE-NEGATIVE-CYCLES$(1, K_1, t)$ was not successful, then t is not added to I_1 and is discarded. We proceed in turn through all indices of I, updating the sets I_1 and K_1 as appropriate. The set $\overline{I}$ keeps track of those indices already processed.

Algorithm 9 CYCLE-CONSTRUCTION$(\widetilde{\mathcal{P}}, \preceq)$

```
    r ← min_≼{i ∈ I}
    Ī ← {r}
 3: I_1 ← {r}
    K_1 ← NULL
    while I \ Ī ≠ ∅ do
 6:   t ← min_≼{i ∈ I \ Ī}
      Ī ← Ī ∪ {t}
      K_2 ← ELIMINATE-NEGATIVE-CYCLES(1, K_1, t)
 9:   if K_2 ≠ NULL then
        I_1 ← I_1 ∪ {t}
        K_1 ← K_2
12:   end if
    end while
    return I_1, K_1
```

Theorem 5.2.2 *Let $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ and the total order $\preceq$ on I be given. Let I_1, K_1 be the sets returned by* CYCLE-CONSTRUCTION$(\widetilde{\mathcal{P}}, \preceq)$ *and define $\widetilde{\mathcal{P}}_1 = \{\widetilde{P}_i \mid i \in I_1\}$. Then the network $G(\widetilde{\mathcal{P}}_1, \preceq, K_1)$ has no negative cycles.*

Proof: Suppose I_1 and K_1 are returned by CYCLE-CONSTRUCTION$(\widetilde{\mathcal{P}}, \preceq)$. If $I_1 = \{r\}$ and $K_1 =$ NULL then $G(\widetilde{\mathcal{P}}_1, \preceq, K_1)$ has only one node (and no cycles). Otherwise, no index t was added to I_1 without first finding a corresponding K_1 such that $G(\widetilde{\mathcal{P}}_1, \preceq, K_1)$ has no negative cycles. ■

5.2.1 Implementation

Implementing the cycle heuristic does not require any complicated data structures except those used by the routine FIND-NEGATIVE-CYCLE(v). This routine is discussed in Section 5.2.1.1 and utilizes

well-known and documented routines, all of which run in time $O(|I_u|^3)$. The running time of the cycle heuristic is pseudo-polynomial since the selected k_{it} is increased by one unit for each execution. The routine ELIMINATE-NEGATIVE-CYCLES(u, K_u, t) may change each of the $|I_u|$ k_{it} values as many as $\max\{n_i, n_t\}$ times. If we define $n = \max_{i \in I}\{n_i\}$ then the total number of times the **while** loop in ELIMINATE-NEGATIVE-CYCLES(u, K_u, t) can be executed is $O(n|I_u|)$. Inside that loop, we need to find a negative cycle, which takes time $O(|I_u|^3)$, and walk the cycle to find (i, t) and (t, j), which takes time $O(|I_u|)$. Thus, the total time taken by the **while** loop is $O(n|I_u|^4)$ which outweighs the initialization cost of the routine. The main CYCLE-CONSTRUCTION$(\widetilde{\mathcal{P}}, \preceq)$ routine executes the ELIMINATE-NEGATIVE-CYCLES(u, K_u, t) routine $|I|$ times. This gives us a total running time of $O(n|I|^5)$.

5.2.1.1 Finding Negative Cycles

There are many different ways to find negative cycles in a network $G = (N, A)$ if they exist. In fact most shortest path algorithms include a negative cycle detection routine to ensure that the shortest path distance labels they output are valid. For example the Bellman-Ford algorithm [9], after relaxing the distance labels on each arc $|N|$ times, checks the shortest path optimality conditions. If these are violated, then at least one negative cycle exists in G. An offending cycle can be extracted using the predecessor array constructed by the algorithm.

Another choice for finding negative cycles in G is the FIFO label-correcting algorithm [1]. This algorithm can detect a negative cycle during its operation by ensuring that the predecessor array represents a tree (i.e., is cycle-free). If at any point a cycle is detected using the predecessor array, then at least one negative cycle exists in G. In fact, the cycle detected by the predecessor array is one such negative cycle.

Alternatively, one can use the Floyd-Warshall algorithm [1] to find negative cycles in G. During the algorithm's operation, we monitor the diagonal entries of the distance matrix (the $d[i, i]$ entries). If at any point one of these entries becomes negative, we know that at least one negative cycle exists in G and we can terminate the algorithm at the end of the current iteration. The algorithm's predecessor matrix can then be used to extract a negative cycle (if one exists).

In our cycle heuristic, it may be desirable to find the most negative weight cycle and correct it first. Unfortunately, finding a cycle of minimum weight in a general network G is NP-hard [12]. However we can find in polynomial time a cycle of minimum mean weight [1]: i.e., a cycle C that minimizes $\frac{w[C]}{|C|}$. If this cycle has negative mean weight then it has negative total weight. On the other hand, if a cycle with minimum mean weight has non-negative mean weight, then there must be no cycle in G with negative total weight. Note that the network $G(\widetilde{\mathcal{P}}, \preceq, K)$ is complete and thus all of these cycle-detecting algorithms run in time $O(|I|^3)$. The experimental results obtained using these different routines to find negative cycles are presented in Chapter 6.

5.2.1.2 Choosing a Total Order

Another decision that has to be made during the implementation of the cycle heuristic is that of creating the total order $\preceq$. This decision matters as it determines the entering pattern at each iteration of the heuristic. The simplest way to do this would be to order the patterns randomly. However some improvements can be obtained if the patterns are ordered deliberately. One approach is to sort the patterns by their usage ratio $\frac{n_i d_i}{B}$. The approach that produced the most success in our experiments, however, involved selecting an entering index in the same way as the Prim heuristic. That is, the entering index t was selected using

$$t = \operatorname*{argmin}_{j \in I \setminus \overline{I}} \Big\{ \min_{i \in I_1} \{B'_{ij}\} \Big\}.$$

This ordering is best determined during run-time using a priority queue in a similar method as implemented in the Prim heuristic. This approach does not change the asymptotic running time of the heuristic. An experimental comparison of different orderings is presented in Chapter 6.

5.2.1.3 Modifications for General Patterns

The major advantage of the cycle heuristic is that, although we have designed it to solve the Linearized Maximum Template problem, at its core it is simply a heuristic to eliminate negative cycles from a network with variable arc lengths. Because of this, it can not only be used to eliminate negative cycles from the network $G(\widetilde{\mathcal{P}}, \preceq, K)$, but from the network $G(\mathcal{P}, \preceq, K)$ as well. In fact, because of the similarities between patterns and linearized patterns, the only thing that needs to be changed is the routine FIND-NEGATIVE-CYCLE(v), which should return a negative weight cycle in

the network $G(\mathcal{P}_v, \preceq, K_v)$ where $\mathcal{P}_v = \{P_i \mid i \in I_v\}$. Everything else about the heuristic can remain the same and it will output I_1 and K_1 so that valid starting positions for $\mathcal{P}_v$ can be obtained from shortest paths from (or to) some given root node r in the network $G(\mathcal{P}_1, \prec, K_1)$.

5.3 Bin-Based Heuristics

Recall that in Section 2.1 we made a connection between the Minimum Templates Problem and the well-known Bin Packing Problem. The Bin Packing Problem in its simplest form takes a set of items I and finds a partition of I into disjoint sets $\mathcal{I} = \{I_u \mid u \in U\}$ such that some feasibility constraint is satisfied for each set (or bin). In Chapter 2 we claimed that what separated the problem of distributing patterns to templates from other bin packing instances, was that it is difficult to determine whether adding a pattern to an existing template violates that template's feasibility. In fact, as we saw in Chapter 4, this problem is in itself NP-hard. However, we have at this point developed heuristics which attempt to decide whether a pattern will fit into an existing template. Such a heuristic can be used as a subroutine for a standard bin packing procedure to find approximate solutions for the Minimum Templates Problem.

Algorithm 10 FIRST-FIT(I)

```
    I ← ∅
    n ← 0
 3: f ← FALSE
    for all t ∈ I do
      u ← 1
 6:   while u ≤ n and f = FALSE do
        if FIT(u, t) then
          I_u ← I_u ∪ {t}
 9:       f ← TRUE
        end if
        u ← u + 1
12:   end while
      if f = FALSE then
        n ← n + 1
15:     I_n ← {t}
        I ← I ∪ {I_n}
      end if
18: end for
    return I
```

A common heuristic for the Bin Packing Problem is known as the First-Fit heuristic [8]. A general routine for this strategy is shown in Algorithm 10. It starts with an empty collection of bins. Then, for each element $t \in I$ it tries to fit t in one of the existing bins in $\mathcal{I}$. The routine FIT(u, t) is used to determine whether t can fit feasibly in bin I_u. The first time this routine returns TRUE, t is added to I_u and the heuristic moves on to the next element of I. If no bin I_u is found such that FIT(u, t) returns TRUE then a new bin is created solely for t and is added to the collection of bins $\mathcal{I}$.

This heuristic can be modified for use with either the Prim heuristic in the case of linearized patterns, or the cycle heuristic in the case of either general patterns or linearized patterns. For the Prim heuristic, this involves using the routine PRIM-CHECK-ALL-FEASIBILITY(t) in place of the FIT(u, t) routine in line 7 of the First-Fit heuristic, and then maintaining the correct zero-feasible (or contiguous) ordering when adding t to I_u. Finally, for the cycle heuristic, we need to maintain the set K_u for each bin I_u and then we can use the routine ELIMINATE-NEGATIVE-CYCLES(u, K_u, t) in place of the FIT(u, t) routine.

A final modification to the standard First-Fit heuristic that has performed well in our experience[9] is to sort the bins in order of increasing number of patterns. This way, each pattern t goes into the bin with the fewest patterns selected from those bins in which it can feasibly fit. This sorting needs to be done once for each execution of the **for** loop in FIRST-FIT(I).

[9]See Chapter 6 for detailed experimental results.

Chapter 6

Computational Results

We now discuss the empirical performance of the heuristics developed in Chapter 5. We will evaluate their performance under a number of different test scenarios, where they will be compared with each other and also with some competing approaches. These latter approaches include a heuristic developed by Korst, et al. [17] (which we will refer to as the Korst heuristic) and the Mixed Integer Programming (MIP) formulations given in Comment 3.1.13. Before presenting any results, however, we discuss how the test problems were generated.

6.1 Generating Test Problems

We decided to focus our attention on testing these heuristics for linear patterns. This allows us to consider heuristics that only work on linearized patterns (i.e., Prim and Kruskal) and to compare these against heuristics for the equivalent Periodic Scheduling Problem. Consequently, almost all of the results presented will be for linear patterns, with the exception of Section 6.3 which deals with the industrial data set that first inspired this research.

Fitting together two linear patterns $\widetilde{P}_i$ and $\widetilde{P}_j$ with $n_i = n_j = n$ is relatively easy. In fact, $\widetilde{P}_i$ and $\widetilde{P}_j$ will fit together iff $\dfrac{B}{n} \geq d_i + d_j$. Moreover, if a large set of linear patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ includes a subset of patterns $\widetilde{\mathcal{P}}_1 = \{\widetilde{P}_i \mid i \in I_1\}$ in which $\widetilde{P}_i = (B, n, d_i)$ for all $i \in I_1$ and $\dfrac{B}{n} \geq \displaystyle\sum_{\ell \in I_1} d_\ell$, then the entire subset $\widetilde{\mathcal{P}}_1$ can be replaced with a single linear pattern $\widetilde{P}_1 = \left(B, n, \displaystyle\sum_{\ell \in I_1} d_\ell\right)$. Because of

this simplicity, we decided that our test data should contain no pairs of patterns $\widetilde{P}_i$ and $\widetilde{P}_j$ with $n_i = n_j$.

Moreover, patterns $\widetilde{P}_i$ and $\widetilde{P}_j$ with $n_i \mid n_j$ are also relatively easy to fit together as shown in Lemma 2.2.1. In fact the entire approach to testing whether two patterns fit together or not involves reducing any two patterns to the case of $\widetilde{P}_j$ and $\widetilde{P}_{ij}$ where $n_j \mid [n_i, n_j]$. Additionally, the heuristic developed by Korst, et al. [17] is specifically designed to exploit divisibilities (including multiplicities) amongst the n_i's. Thus, because this ground has already been covered, we decided to evaluate our heuristics on sets that contain no patterns $\widetilde{P}_i$ and $\widetilde{P}_j$ such that $n_i \mid n_j$.

Recall that two patterns $\widetilde{P}_i$ and $\widetilde{P}_j$ cannot be fit on the same template if $B_{ij} < d_i + d_j$. Thus, if we have a set of patterns containing two such patterns, then the Linearized Feasible Fit problem (LinFit) can be answered quickly. Also some preprocessing techniques [17] can be carried out on a set of patterns $\widetilde{\mathcal{P}}$ before solving the Linearized Minimum Templates problem; these can speed up solution times and provide robust lower bounds on the optimal solution. To remove the influence of preprocessing and to make the problems more challenging to solve, we only consider sets of patterns such that $B_{ij} \geq d_i + d_j$ for all pairs. Additionally, enlarging B for all patterns enlarges B_{ij} for all pairs of patterns. As discussed in Section 5.1, larger B_{ij}'s make patterns easier to fit together. Consequently we will design test sets such that B is as small as feasibly possible.

Our objective then is to generate a set of patterns $\widetilde{\mathcal{P}} = \{\widetilde{P}_i \mid i \in I\}$ that would achieve all of these aims. For the Linearized Feasible Fit tests, we produced 10,000 replicated sets with m patterns, where $m = |I| \in \{3, 4, ..., 10\}$. To generate each such set, we performed the following steps:

- Create the set $\mathcal{N} = [3, 100] \cap \mathbb{Z}$ of possible n_i values.
- For $i = 1..m$:
  - randomly select n_i from $\mathcal{N}$,
  - remove all factors and multiples of n_i from $\mathcal{N}$.
- For $i = 1..m$ randomly select d_i from $[1, 30] \cap \mathbb{Z}$.

- Set $B = \max_{i \in I} \{(d_i + d_j)[n_i, n_j]\} + 1$[1]

This procedure was repeated 10,000 times to generate sets each having m patterns. We call this complete data set the *Small Pattern-Set Collection*.

In addition, we wanted sets containing a large number of patterns to test heuristics for solving the Linearized Maximum Template problem (LinMaxTemp) and the Linearized Minimum Templates problem (LinMinTemp). Here the number of patterns m in each set ranges from 25 through 250 patterns in increments of 25. Because of the increase in running times for these large sets, we could no longer generate such a large number of replications. As a result, we only generated 10 sets of patterns for each set size m. The values n_i were generated as described earlier, except $\mathcal{N}$ was initialized as $[3, 1000] \cap \mathbb{Z}$. We will call this complete data set the *Large Pattern-Set Collection*.

All random data was generated using the Mersenne Twister Algorithm [21]. All tests were run on an Intel Pentium E2140 1.6GHz CPU with 2 GiB of RAM. The machine was running Ubuntu Linux 11.10 64-bit. Code was written in Java and run using the OpenJDK 1.6.0_23 64-bit virtual machine.

6.2 Evaluation of Heuristics

We begin our analysis of the heuristics presented with a comparison of the different algorithmic options used. In particular, we first consider approaches for solving the Linearized Feasible Fit problem.

6.2.1 Prim Heuristic - Starting Positions

As discussed in Section 5.1.3.1 the quality of the solutions produced by the Prim heuristic is sensitive to how we choose the initial index r in the template I_1. The different initialization techniques tried were:

- SMALLEST START: select r to be one of the patterns $p, q \in I$ such that $B'_{pq} = \min_{i,j \in I} \{B'_{ij}\}$.
- RANDOM START: choose a random index $r \in I$.

[1] The extra unit in the length of B is added to prevent computational rounding errors that arose during preliminary testing.

- All Start: try each $r \in I$ in turn and report the best solution found.

The test involved solving the Linearized Feasible Fit problem for the *Small Pattern-Set Collection*, generated as described earlier. We focus here on the number of sets for which the Prim heuristic was able to find a contiguous ordering that included all patterns in the given set: that is, the number of sets for which the Prim heuristic answered the Linearized Feasible Fit problem to the affirmative.

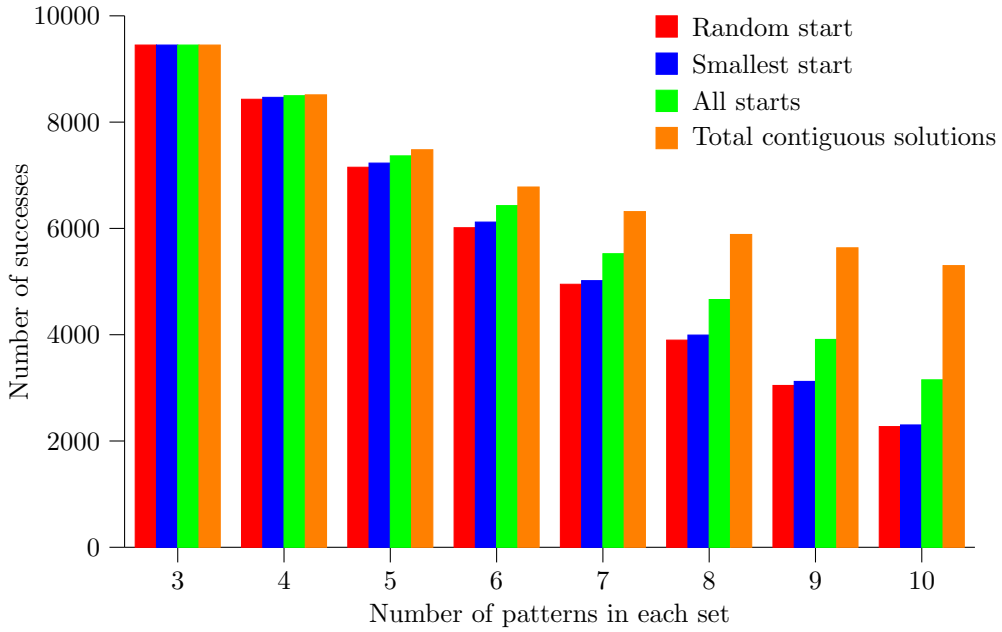


Figure 6.2.1: Success rates of different initialization techniques for the Prim heuristic

These results are displayed in Figure 6.2.1 and are compared with the total number of sets such that a contiguous ordering exists using all patterns in the set; this total number was determined by checking inequality (3.2.6) from Theorem 3.2.7 for every permutation of the patterns. The results indicate that choosing a initial pattern with the smallest B'_{ij} gives somewhat better results than a random choice. However, as would be expected, trying all possible indices r yields the best results. The All Start approach does take longer than the other two approaches, but it still only requires a fraction of a second per set. Trying all possible permutations takes significantly longer, especially once we reach 10 patterns per set. This consideration limited our range for m to a maximum of 10 patterns.

In this experiment only 54% of sets with 10 patterns had a contiguous solution. So even though the ALL START option of the Prim heuristic only found solutions for 32% of the sets with 10 patterns, it did find 60% of all possible contiguous solutions. This leads us to consider the density of zero-feasible and contiguous solutions.

6.2.2 Zero-Feasible vs. Contiguous Orderings

Here we compare the number of generated sets of patterns with zero-feasible solutions to the number of sets with contiguous solutions. We would also like to gauge the effectiveness of the Prim and Kruskal heuristics in finding those solutions. As before, the test involved solving the Linearized Feasible Fit problem for the *Small Pattern-Set Collection*, generated as described earlier.

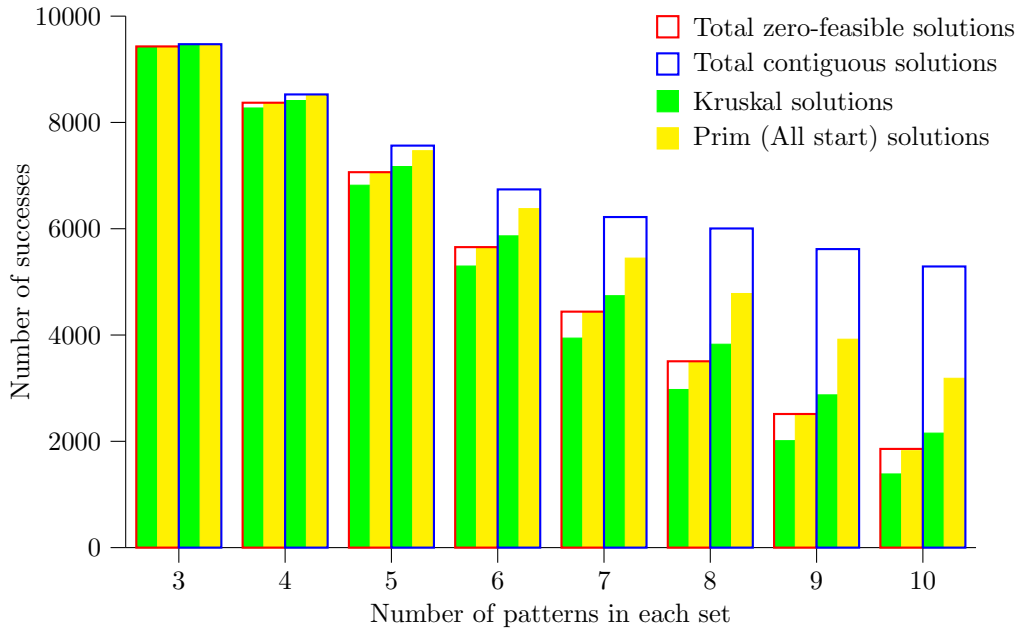


Figure 6.2.2: Density of special orderings and success rates of Prim and Kruskal heuristics

Figure 6.2.2 shows the number of zero-feasible and contiguous solutions found using the brute force approach of testing inequalities (3.2.4) and (3.2.6) respectively for every permutation. The figure also shows the number of each type of solution that the Prim (using the ALL START approach) and Kruskal heuristics were able to find. Additionally, it is evident from Figure 6.2.2 that there are

more sets of patterns that have just a contiguous ordering, than have a zero-feasible ordering; this is more clearly pronounced as the number of patterns m increases. Also, both the Prim and Kruskal heuristics are able to find more contiguous orderings than zero-feasible orderings. However, they are able to find a larger percentage of possible zero-feasible solutions than of possible contiguous solutions as summarized in Table 6.2.1.

Number of patterns per set		3	4	5	6	7	8	9	10
Prim (All start)	% of zero-feasible orderings found	100%	100%	100%	100%	100%	100%	99%	98%
	% of contiguous orderings found	100%	100%	99%	95%	88%	80%	70%	60%
Kruskal	% of zero-feasible orderings found	100%	99%	97%	94%	89%	85%	80%	75%
	% of contiguous orderings found	100%	99%	95%	87%	76%	64%	51%	41%

Table 6.2.1: Percentage of feasible orderings found by Prim and Kruskal heuristics

6.2.3 Cycle Heuristic - Negative Cycle Finders

We now consider the cycle heuristic, which also had several options that could affect the quality of the solutions. The first option we will address is that of the negative cycle finding routine. For this we tested three different options [1]:

- FIFO label-correcting algorithm,
- Floyd-Warshall algorithm, and
- minimum mean cycle algorithm.[2]

All of these negative cycle finding options were tested by solving the Linearized Feasible Fit problem for the *Small Pattern-Set Collection*, generated as described earlier. The cycle heuristic used a random insertion order for all options. Data was collected on the number of sets where the heuristic

[2]See Section 5.2.1.1 for more details on these algorithms.

was able to fit all patterns in the set on the same feasible template. The results, shown in Figure 6.2.3, indicate that in our experiments, the FIFO label-correcting algorithm was a better choice than the Floyd-Warshall algorithm. Additionally, both of these options produce results that are far superior to those produced when using the minimum mean cycle algorithm.

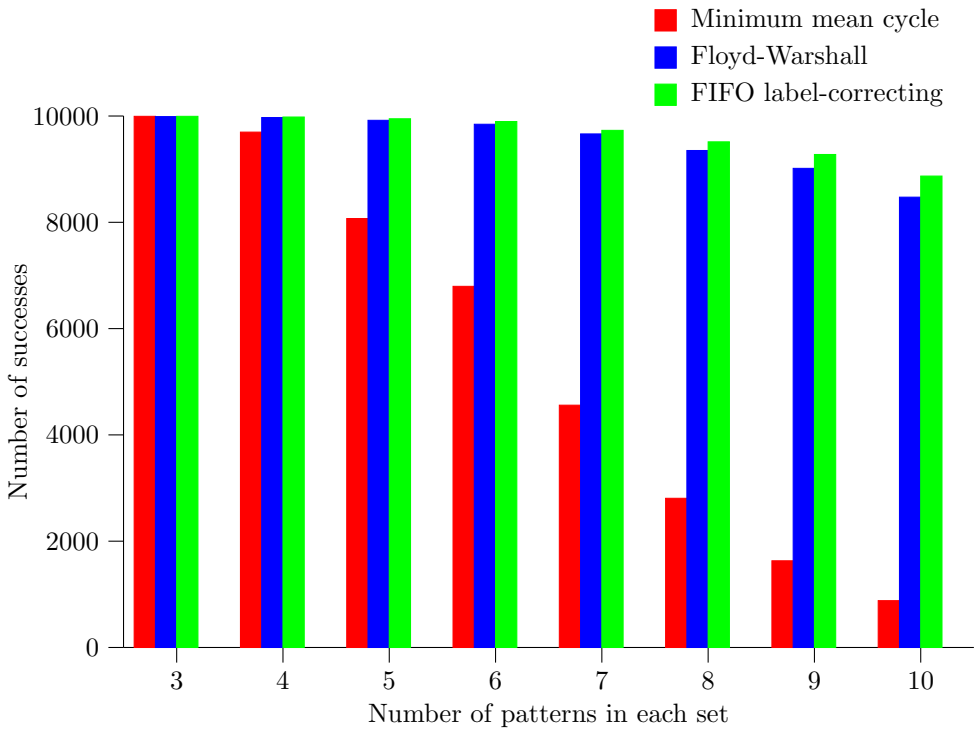


Figure 6.2.3: Success rates of different negative cycle finders for the cycle heuristic

6.2.4 Cycle Heuristic - Insertion Orders

Continuing with the cycle heuristic, we now consider the difference that the insertion order makes on the quality of the solutions. This is the order which determines the choice of entering index t. We considered four options:

- increasing size ratio $\frac{n_i d_i}{B}$,
- decreasing size ratio $\frac{n_i d_i}{B}$ [17],
- random order, and

- Prim heuristic ordering.[3]

As usual, these options were tested by solving the Linearized Feasible Fit problem for the *Small Pattern-Set Collection*, generated as described earlier. The FIFO label-correcting algorithm was used as the negative cycle finder for all tests. The results, displayed in Figure 6.2.4, show that in our experiments, using the Prim order gave the best performance in terms of fitting patterns on a single template. In fact, this combination of options allowed the cycle heuristic to find solutions for no fewer than 98.8% of all sets of size $m \leq 10$. Sorting the patterns by decreasing size also performed well, returning better solutions than the reverse order. Interestingly, even a random order proved better than using the increasing size order.

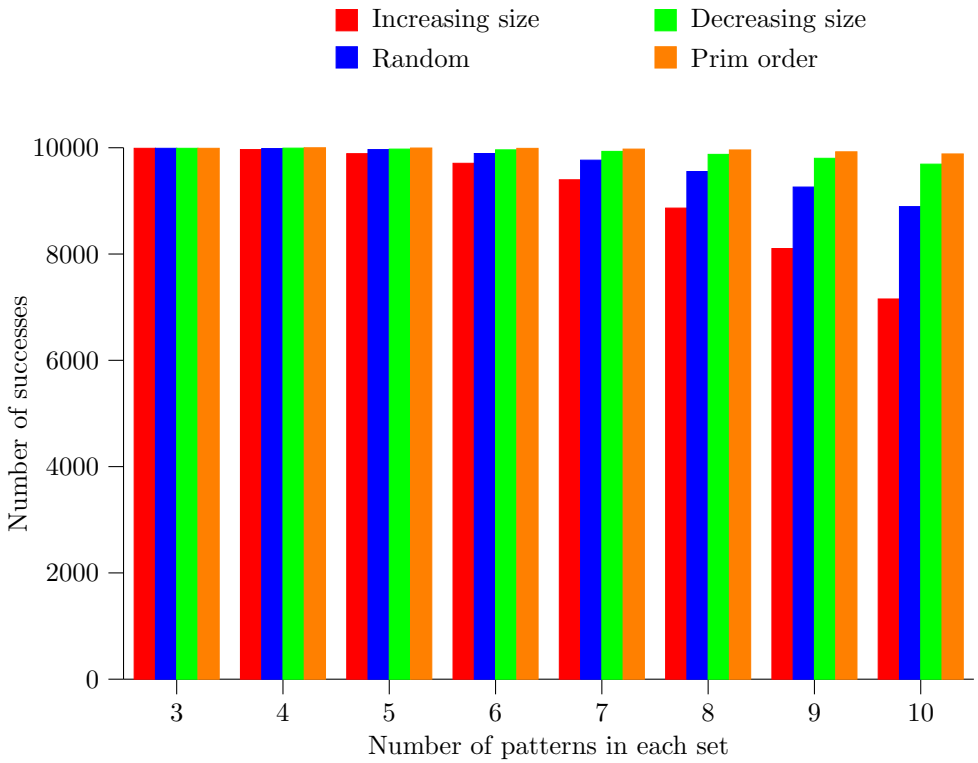


Figure 6.2.4: Success rates of different insertion orders for the cycle heuristic

[3]See Section 5.2.1.2 for more details.

6.2.5 Linearized Feasible Fit Problem

Finally, we wish to compare the best options for our three heuristics head-to-head. These heuristics are:

- the Kruskal heuristic, finding contiguous orderings,
- the Prim heuristic, finding contiguous orderings and trying all possible starts, and
- the cycle heuristic, using the FIFO label-correcting algorithm to find negative cycles and the Prim insertion order.

These three heuristics were again tested by solving the Linearized Feasible Fit problem for the *Small Pattern-Set Collection*, generated as described earlier. In addition, we used Gurobi [14], a commercial mixed integer program solver, to find solutions to $\textsc{Fit}(\widetilde{\mathcal{P}})$, the MIP formulation described in Comment 3.1.13, for each set of patterns in the *Small Pattern-Set Collection*. Gurobi was given a 10 minute time limit to find a solution to each set. It returned solutions for most sets that were deemed feasible for the MIP formulation to machine precision, but unfortunately such solutions did not always represent truly feasible solutions.

The results, displayed in Figure 6.2.5, show that for this experiment, the cycle heuristic is far superior to the other two heuristics, which are hamstrung by the decreasing density of possible contiguous orderings. The results also indicate that the Prim heuristic produces consistently better results for this problem than the Kruskal heuristic. Also, we observe that, while Gurobi returned solutions for most sets of patterns, the vast majority of those solutions were actually infeasible. This shows that, in addition to the normal problems associated with the running time of MIP solvers, Gurobi is unsuited to solving the Linearized Feasible Fit problem formulation because of machine precision issues. Additionally, we were able to modify the Korst heuristic [17] to solve the Linearized Feasible Fit problem, but it was unable to find solutions to a single set.

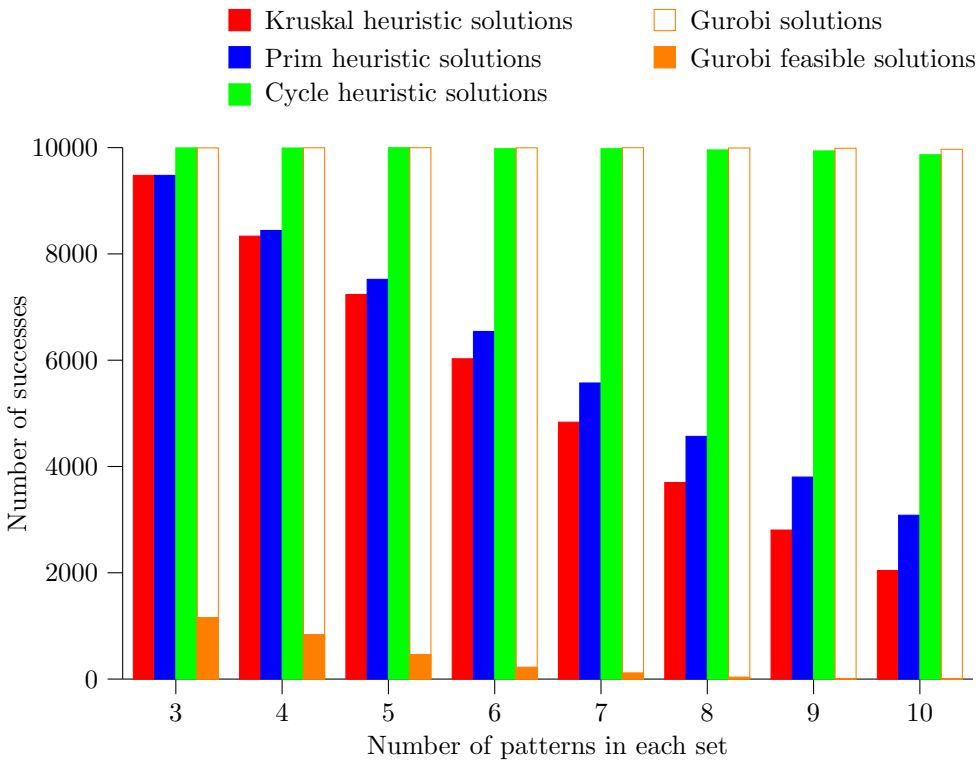


Figure 6.2.5: Heuristic performance for Linearized Feasible Fit problem

6.2.6 Linearized Maximum Template Problem

We now switch our focus to the Linearized Maximum Template Problem. For this problem, we evaluated the following heuristics:

- the Kruskal heuristic, finding contiguous orderings,
- the Prim heuristic, finding contiguous orderings and trying all possible starting indices r,
- the cycle heuristic, using the FIFO label-correcting algorithm to find negative cycles and the Prim insertion order,
- Gurobi, solving the Mixed Integer Programming formulation of the LinMaxTemp problem, with a 60 minute time limit, and
- the Korst heuristic [17].

Each heuristic was tested by solving the Linearized Maximum Template Problem on the *Large Pattern-Set Collection*, generated as described earlier. The results, displayed in Figure 6.2.6, show that surprisingly, the Prim heuristic performs almost as well as the cycle heuristic as the number of patterns per set grows. This may be an indication that, for large numbers of patterns per set, the cycle heuristic faces a saturation problem that the Prim heuristic does not, despite the lessening density of contiguous solutions. It is also noteworthy that, although the solutions produced by Gurobi in 60 minutes are reasonably competitive for sets of size up to 150 patterns, the quality of the solutions obtained actually deteriorates for larger problems. We believe that this is due to the fact that the time taken to solve the linear programming relaxation at each node increases to the point where considerably fewer nodes of the branch-and-bound tree are searched, negatively affecting the solution quality. The results of the Korst heuristic were quite disappointing as no solution with more than 10 patterns in a template were produced.

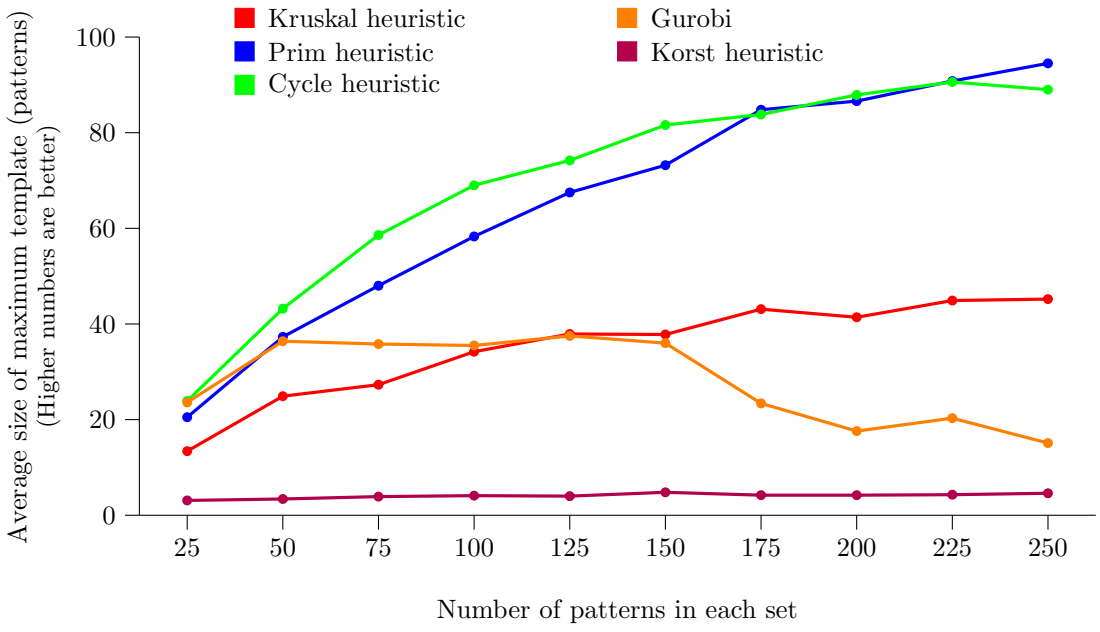


Figure 6.2.6: Heuristic performance for Linearized Maximum Template problem

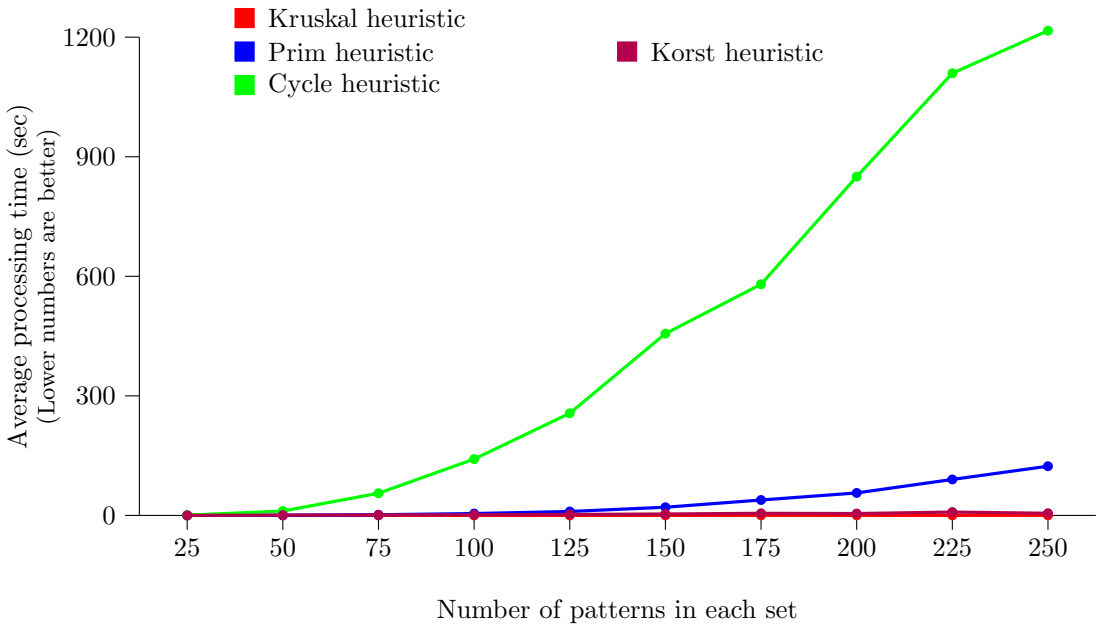


Figure 6.2.7: Heuristic processing time for Linearized Maximum Template problem

Another consideration is the time taken to produce these results. Figure 6.2.7 shows the average processing time (in seconds) taken by each heuristic[4]. While the cycle heuristic does produce better results when there are fewer than 150 patterns per set, the extra time taken is significant. Once the number of patterns per set reaches 175, the Prim heuristic produces comparable results with far less processing time. It should also be noted that, while the quality of the solutions produced by the Kruskal heuristic is not as high, the processing time never exceeds 0.25 seconds, even when processing a set of 250 patterns. Thus, use of the Kruskal heuristic might be desirable in a real-time environment. Note that the Korst heuristic had processing times similar to those for the Kruskal heuristic, so their graphs are hard to differentiate in Figure 6.2.7.

6.2.7 Linearized Minimum Templates Problem

The last problem associated with linear patterns that we consider is the Linearized Minimum Templates problem. For this problem, we used the following heuristics:

- the Kruskal heuristic, finding contiguous orders,

[4]The time taken by Gurobi was always the maximum 3600 seconds allowed, and this value would not fit on the scale of this chart.

- the Modified First-Fit bin packing heuristic, using the contiguous ordering version of the PRIM-CHECK-ALL-FEASIBILITY(t) routine in place of FIT(u, t),
- the Modified First-Fit bin packing heuristic, using the FIFO label-correcting algorithm version of the ELIMINATE-NEGATIVE-CYCLES(u, K_u, t) routine in place of FIT(u, t),
- Gurobi, solving the Linearized Minimum Templates MIP with a 60 minute time limit, and
- the Korst heuristic [17].

Each heuristic was tested by solving the Linearized Minimum Template problem on the *Large Pattern-Set Collection*, generated as described earlier. The results, displayed in Figure 6.2.8, show that the cycle heuristic consistently achieves the smallest average number of templates. Also, though the Prim heuristic has been superior to the Kruskal heuristic for every previous test, here the Kruskal heuristic is clearly better for the problem it was designed to solve (generating a disjoint set of templates). The results for the Korst heuristic are only shown for 25 and 50 patterns per set because it would not reasonably fit on the graph. However the slope shown in Figure 6.2.8 for the Korst heuristic data is indeed maintained once leaving the bounds of the graph. Finally, results for Gurobi are also only shown for 25 and 50 patterns. Recall that this MIP formulation requires an initial upper bound on the number of templates. For each set, we used the Kruskal heuristic's solution to produce an upper bound for the MIP. Unfortunately, once the number of patterns per set reached 75, Gurobi was unable to find a feasible solution for any MIP in the 60 minutes allotted. As before, we believe that this is due to the enormous size of the resulting linear programming relaxation of the MIP.

Once again we consider the processing time of each heuristic. These results, displayed in Figure 6.2.9, show that, as before, the cycle heuristic takes the longest time for processing. However, the Kruskal heuristic again produces reasonable solutions in well under a second. Similar times were achieved by the Prim heuristic which lead to their nearly identical graphs appearing in Figure 6.2.9. Notice that the processing times for all heuristics are significantly shorter than those for the Linearized Maximum Template problem, even for sets of the same size. This occurs because the running time of all heuristics is dependent on the size of the template. Whereas the largest template reached sizes in excess of 50 patterns in the LinMaxTemp problem, here the largest templates are of the order of 10 patterns, significantly reducing the processing time.

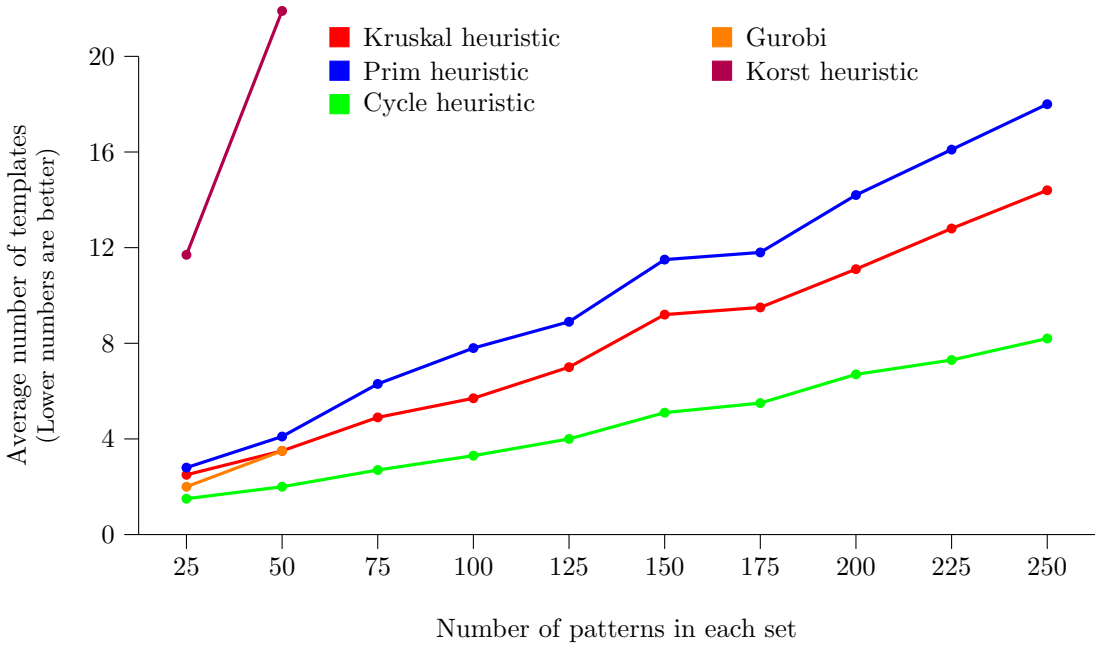


Figure 6.2.8: Heuristic performance for Linearized Minimum Template problem

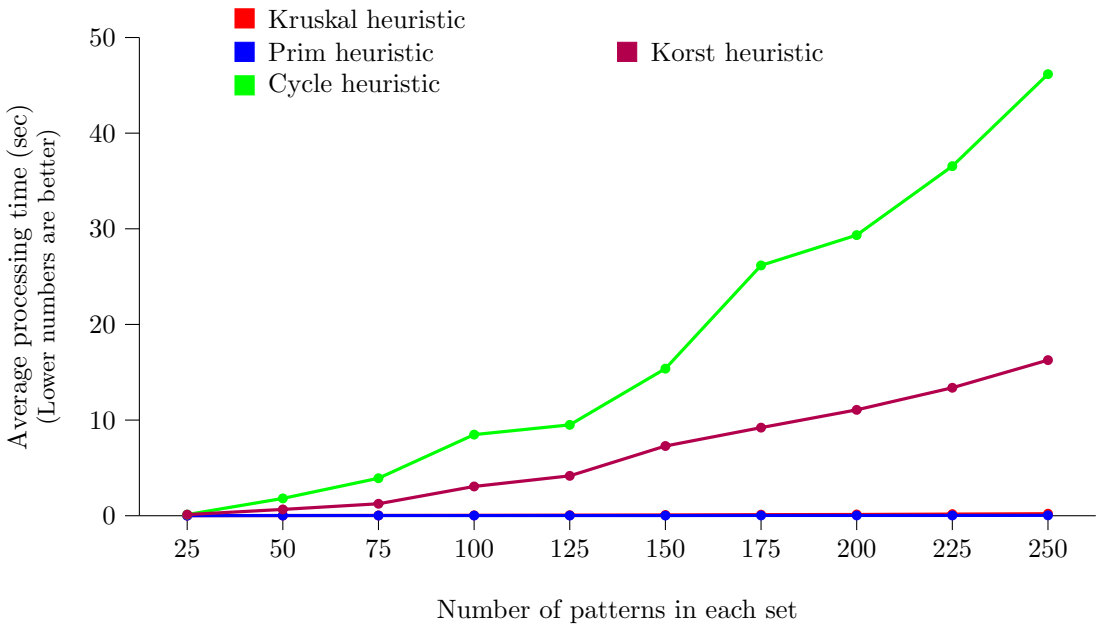


Figure 6.2.9: Heuristic processing time for Linearized Minimum Template problem

6.2.7.1 Comparison to the Korst Heuristic

The Korst heuristic [17] has not performed well in the experiments presented so far. This is mainly due to the fact that we specifically constructed data sets that did not possess the qualities that the Korst heuristic was designed to exploit. In fact our heuristics are competitive, even when used in an experimental setting similar to that used by Korst [17] to show the effectiveness of their heuristic. This experiment was designed to solve the Periodic Scheduling Problem (PSP) for 100 sets of m tasks where the period $p(i)$ of each task $i \in \{1, ..., m\}$ is selected uniformly from $[1, p_{\max}] \cap \mathbb{Z}$. The processing time $e(i)$ of task i is then selected uniformly from $[1, p(i)] \cap \mathbb{Z}$.

Recall that, as shown in Chapter 3, the Periodic Scheduling Problem is equivalent to the Linearized Minimum Templates problem. Thus, all heuristics for the Linearized Feasible Fit problem can be used to solve PSP instances. To do this, we calculate $B_{ij} = \big(p(i), p(j)\big)$ and set $d_i = e(i)$. Thus, we were able to generate a similar data set using the same parameters as in [17] to test our heuristics. We also implemented some of the preprocessing ideas from [17] before running our heuristics. We then used the Modified First-Fit bin packing heuristic, with the contiguous ordering version of the Prim-Check-All-Feasibility(t) routine in place of Fit(u, t), to solve the PSP instances in this data set[5].

The results presented in [17] are given as the average percentage by which the heuristic exceeded a calculated lower bound. These results are reproduced in Table 6.2.2 while the equivalent results from our experiment are shown in Table 6.2.3. The two entries highlighted in red represent the largest deviation between the two tables, where the Korst heuristic slightly improves upon the Modified First-Fit heuristic. Thus, not only do our heuristics represent a significant improvement upon that of Korst [17] when dealing with sets of patterns with no multiplicities among the n_i's, but they are competitive with that heuristic even when multiplicities do arise.

[5]The cycle heuristic was also used in combination with the Modified First-Fit heuristic in this experiment and yielded results similar to those seen here.

	$p_{\max}$									
m	10	20	30	40	50	60	70	80	90	100
10	0.55	0.43	0.41	0.27	0.00	0.29	0.13	0.39	0.00	0.00
20	1.23	0.93	0.89	0.46	0.50	0.27	0.24	0.36	0.32	0.55
30	2.29	1.61	1.21	1.19	0.56	0.51	0.54	0.54	0.33	0.50
40	2.29	1.72	1.50	1.22	1.19	1.17	0.93	0.56	0.85	0.50
50	2.70	2.50	1.76	1.70	1.22	1.35	1.29	0.91	0.52	0.86
60	3.48	2.69	2.07	1.64	1.44	1.34	1.16	1.00	0.78	0.88
70	3.86	3.28	2.89	2.01	2.06	1.78	1.19	1.20	1.03	0.62
80	4.20	3.71	2.84	2.44	2.22	1.62	2.18	1.26	1.14	1.03
90	3.95	4.04	3.90	3.02	2.46	1.99	1.31	1.54	1.15	1.34
100	4.16	4.63	3.77	3.09	2.32	2.16	1.92	1.59	1.89	1.52
110	4.16	4.42	4.14	3.23	2.69	2.42	2.21	1.88	1.37	1.51
120	4.43	5.10	4.45	3.79	3.43	2.53	2.75	2.17	1.88	1.94
130	4.53	4.78	4.51	4.02	2.94	2.76	2.95	2.10	2.39	1.81
140	4.33	5.02	4.67	4.70	3.79	3.42	3.20	2.18	2.01	1.78
150	4.39	5.15	4.90	4.48	3.77	3.37	3.13	2.57	2.64	2.11
160	4.40	5.42	5.48	4.63	4.14	3.58	2.84	3.07	2.27	2.29
170	4.60	5.45	5.63	5.03	4.23	3.66	3.49	3.01	2.77	2.47
180	4.53	5.71	5.55	5.39	5.02	4.11	3.78	3.05	2.49	2.86
190	4.24	5.44	6.09	5.36	4.17	4.44	3.89	3.61	3.57	3.08
200	4.24	6.10	5.77	5.27	4.73	4.01	4.04	3.49	3.10	2.84

Table 6.2.2: Periodic Scheduling Problem: Korst heuristic results

	$p_{\max}$									
m	10	20	30	40	50	60	70	80	90	100
10	0.25	0.17	0.39	0.00	0.14	0.31	0.00	0.00	0.00	0.00
20	0.99	0.79	0.72	0.39	0.21	0.18	0.09	0.39	0.26	0.00
30	1.09	1.17	1.02	0.73	0.57	0.42	0.21	0.37	0.16	0.36
40	1.97	2.00	1.51	1.20	0.59	0.49	0.82	0.53	0.40	0.54
50	2.06	1.95	1.64	1.31	1.05	1.02	0.73	0.58	0.71	0.39
60	2.71	2.56	1.63	1.52	1.26	1.31	0.72	0.88	0.92	0.90
70	2.73	2.69	2.55	2.08	1.53	1.39	1.47	1.03	1.08	0.87
80	3.08	2.76	3.05	1.86	2.11	2.15	1.34	1.45	1.10	1.27
90	3.02	3.21	3.24	2.63	2.25	1.98	1.79	1.84	1.25	1.36
100	3.21	3.70	3.40	3.34	2.56	2.29	2.05	1.73	1.58	1.39
110	3.55	3.85	4.15	3.04	3.00	2.79	2.13	2.21	1.51	1.47
120	3.44	3.96	4.50	3.75	3.09	3.23	2.52	2.72	1.86	1.81
130	3.56	4.34	4.46	3.71	3.39	3.49	2.66	2.66	2.27	2.35
140	3.65	4.63	4.52	3.93	3.81	3.34	2.70	2.36	2.41	2.29
150	3.29	4.36	4.87	4.55	3.97	3.47	3.00	2.86	2.74	2.21
160	3.62	4.95	5.14	4.60	3.99	3.65	3.51	3.07	2.85	2.46
170	3.67	5.12	5.88	5.08	4.27	3.95	3.51	3.17	2.87	2.81
180	3.28	4.90	5.64	5.03	4.68	4.35	3.70	3.22	3.45	3.15
190	3.47	5.24	5.60	5.39	4.64	4.59	3.79	3.92	3.21	3.09
200	3.63	5.08	5.52	5.83	5.18	4.67	4.21	3.69	3.71	2.91

Table 6.2.3: Periodic Scheduling Problem: Modified First-Fit (Prim) heuristic results

6.3 Industrial Example

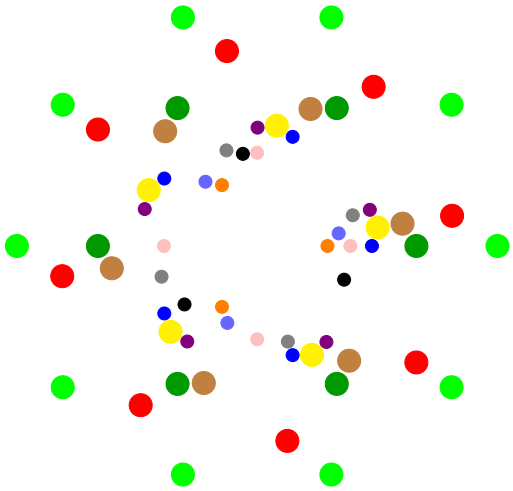
Finally, since we have extended our cycle heuristic to solve problems involving sets of general patterns $\mathcal{P}$, we were able to apply this heuristic to a data set supplied by the industry that first motivated this research. This data set is comprised of sixty-three patterns defined by values in the following sets

$$R_i \in [47.625, 167.5]$$
$$n_i \in \{3, 4, 5, 6, 7, 8, 10\}$$
$$\rho_i \in \{4.5, 8\}.$$

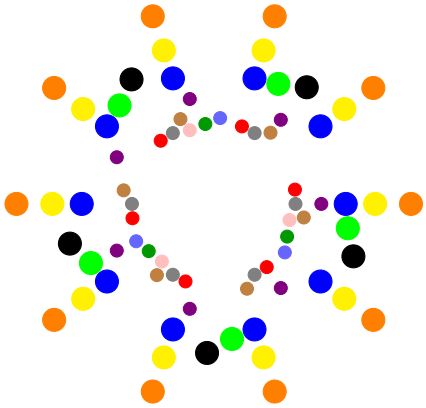
We seek a minimum number of feasible templates that will hold all sixty-three patterns in the data set. To achieve this, we applied the Modified First-Fit bin packing heuristic to this data set using the cycle heuristic to determine bin fit. The cycle heuristic used the FIFO label-correcting algorithm to find negative cycles in $G(\mathcal{P}, \preceq, K)$. This combination found a solution using five templates for the sixty-three patterns in under 0.1 seconds. This solution is shown in Figure 6.3.1. Notice that the patterns are fairly equitably distributed amongst the templates. This is not unexpected as our modifications to the First-Fit heuristic encourage this outcome. Namely, we always try to put a new pattern into a template with the fewest patterns first. This avoids the scenario where one template contains most of the patterns. This equitability feature may be desirable if, for example, we are trying to minimize the number of times workers would have to change these templates.

By comparison, we were able to use the Mixed Integer Programming formulation presented in Section 2.3.2 to attempt to find a solution for this same data set using the Gurobi commercial MIP solver. Recall that this formulation requires an upper bound on the number of templates for efficient solution. When given an upper bound of five templates, Gurobi is able to find such a solution in about 220 seconds[6]. However, it was not able to reduce the number of templates even when given two weeks of uninterrupted processing time. As a result, we feel that our heuristic is very competitive with the commercial alternatives.

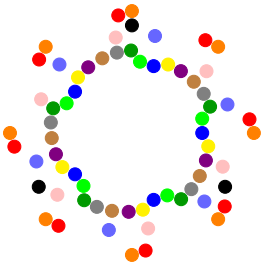
[6]By comparison, when given an upper bound of 6 templates, it finds a solution with 5 templates after 1,142 seconds, and when given an upper bound of 7 templates it takes 8,762 seconds (about 2.4 hours) to find the same 5 template solution.



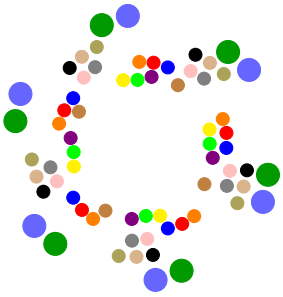
6.3.1(a): Template #1: 12 patterns



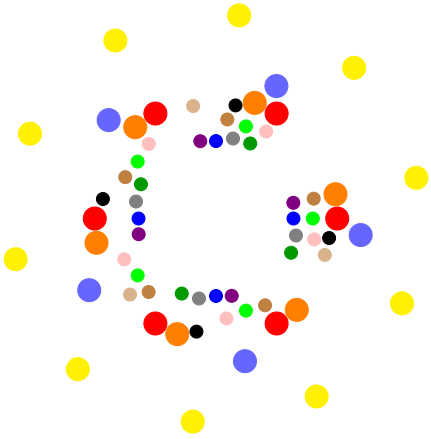
6.3.1(b): Template #2: 12 patterns



6.3.1(c): Template #3: 12 patterns



6.3.1(d): Template #4: 14 patterns



6.3.1(e): Template #5: 13 patterns

Figure 6.3.1: Industrial example solution

Chapter 7

Conclusions

In this dissertation we examined three related geometric packing problems (initially motivated by an industrial situation), derived conditions for the problems' feasibility and established their computational complexity. We were then able to connect those feasibility constraints with a network formulation. If we can eliminate negative cycles from this parameterized network, we are able to find shortest paths from (or to) any given node in the network, and these shortest path distances will yield feasible packings.

We then considered an approximation of the geometric packing problem and transferred most of the previous results, including the network formulation, to this approximation. Two special cases of the approximation, zero-feasible and contiguous orderings, were then analyzed, and we were able to provide feasibility conditions for the existence of these orderings. Additionally, we proved the computational complexity of finding zero-feasible orderings. We further showed that this approximate packing problem is equivalent to the Periodic Scheduling Problem. This equivalence allows some of our more general developments – such as the k_{ij} bounds from Theorem 3.1.12, the network formulation, and the special orderings – to be applied to the many problems arising in the periodic scheduling literature.

Moreover, we were able to exploit two special orderings and our network formulation to design heuristics for the solution of these provably difficult packing problems. These heuristics can also be applied to existing periodic scheduling examples and, in the case of the cycle heuristic, to the problem

of packing general geometric patterns. The heuristics performed favorably when compared with other heuristics developed for the Periodic Scheduling Problem and a state-of-the-art commercial mixed integer program solver.

7.1 Future Work

Additional avenues of research on this topic still remain to be pursued. Some of these are theoretical and include establishing the complexity of obtaining a contiguous ordering. On the other hand, there are several options to consider for improving the heuristics we have developed. For example, the Prim and Kruskal heuristics were motivated by seeking zero-feasible orderings and many of their design decisions were made with this in mind. These heuristics were only later adapted in a rudimentary way to contiguous orderings. This fact is evident when you consider the percentage of zero-feasible orderings that they are able to discover, compared with the percentage of contiguous orderings they can find[1]. We believe that further modifications to the fundamental workings of these heuristics can improve their performance overall, especially considering the larger number of available contiguous orderings.

There are also changes that can be considered for implementing the cycle heuristic. When this heuristic inserts a new index t into a feasible template, it uses a very simple heuristic to find integer values k_{it} for all i in the current template such that the expanded template is also feasible. This problem can in fact be formulated as an integer program with two variables per constraint. Such integer programs are known as 2VIP problems [15] and there are existing pseudo-polynomial algorithms that will solve them exactly. Since the ELIMINATE-NEGATIVE-CYCLES(u, K_u, t) routine is already pseudo-polynomial, it would be useful to know how the running times of the 2VIP algorithms compare to that for the heuristic currently used in ELIMINATE-NEGATIVE-CYCLES(u, K_u, t). More importantly, it would be valuable to assess how using the exact solution at each iteration would affect the quality of the solutions produced by CYCLE-CONSTRUCTION$(\widetilde{\mathcal{P}}, \preceq)$ and the Modified First-Fit heuristic.

Another area of possible future study is that of bin equitability. When solving a bin packing problem

[1] See Table 6.2.1 for supporting experimental data.

such as the Minimum Templates problem, we only insist that our heuristic maintain the feasibility of each bin. However, as results from the industrial example show, it may be desirable to have the patterns evenly distributed across the templates so that workers do not have to change them as often. We believe that our heuristics could be modified to produce solutions that distribute the patterns more evenly over the templates. Moreover, this line of study could be extended to the more general Bin Packing Problem as well.

In this research we have successfully developed construction heuristics that build feasible solutions to the problem of packing patterns into templates. However, it is well known that construction heuristics can make decisions early in the construction that later limit choices to undesirable options. A common approach to correct this situation is the development of improvement heuristics. These take an existing feasible solution and attempt to improve it by suitable interchanges. In our case, this approach should be explored for the problem of finding special orderings, as well as the more general problem of packing patterns into templates. Improvement heuristics could also be developed that focus on improving the equitability of distribution among templates.

Korst, et al. [17] introduced the notion of an overlap graph for a set of periodic tasks (or equivalently, linear patterns). This graph has a node for each task (pattern) and an edge between them iff the constraint from Corollary 3.1.11 holds for those two tasks (patterns). They then find a maximal independent set in that graph and point out that the size of this set provides a lower bound on the number of servers needed to feasibly accomplish all tasks (since no two tasks in the set can fit together). We have elsewhere [24] developed a condition similar to that in Corollary 3.1.11 for determining the existence of valid starting positions for sets of three patterns. It would be of interest to study whether this constraint could be added to the overlap graph to improve the quality of the resulting lower bound.

Appendices

Appendix A Proof of Theorem 2.2.4

For clarity, we restate Theorem 2.2.4:

Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given. Then $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is feasible iff there exists $k \in \mathbb{Z}$ such that

$$k\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - \delta_{ij}.$$

Moreover, if $\delta_{ij} > 0$ for $i \prec j$ then k is unique.

Proof: Let patterns $P_i = (R_i, n_i, \rho_i)$ and $P_j = (R_j, n_j, \rho_j)$ be given. First note that if $|R_i - R_j| \geq \rho_i + \rho_j$ then $\delta_{ij} = 0$, so the requirement is that there exists some k such that

$$k\beta_{ij} \leq \varphi_j - \varphi_i \leq (k+1)\beta_{ij}$$

which in fact holds for all φ_i, φ_j. This aligns with the results of Theorem 2.2.3. Thus, for the rest of this proof we will assume that $|R_i - R_j| < \rho_i + \rho_j$.

Recall that we have assumed that $n_i > 1$ for all $i \in I$, thus $[n_i, n_j] \geq 2$ for all $i \neq j \in I$. Define $P_{ij} = \big(R_i, [n_i, n_j], \rho_i\big)$ as before. Recall that Lemmas 2.2.1 and 2.2.2 together imply that $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is a feasible template iff $P_{ij}(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j, 0) = \emptyset$.

($\Rightarrow$)

Assume that $\{P_i(\overline{\varphi}_i), P_j(\overline{\varphi}_j)\}$ is feasible. Thus $P_{ij}(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j, 0) = \emptyset$. Let

$$k = \left\lfloor \frac{\overline{\varphi}_j - \overline{\varphi}_i}{\beta_{ij}} \right\rfloor.$$

Thus

$$k\beta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i < (k+1)\beta_{ij}.$$

Now if $P_{ij}(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j, 0) = \emptyset$, then $P_{ij}(\overline{\varphi}_i, \ell) \cap P_j(\overline{\varphi}_j, 0) = \emptyset$ for all $\ell \in \mathbb{Z}$. Specifically, $P_{ij}(\overline{\varphi}_i, k) \cap P_j(\overline{\varphi}_j, 0) = \emptyset$. Thus the distance between the center of $P_{ij}(\overline{\varphi}_i, k)$, $(R_i, \overline{\varphi}_i + k\beta_{ij})$, and the center of

$P_j(\overline{\varphi}_j, 0)$, $(R_j, \overline{\varphi}_j)$, must be at least $\rho_i + \rho_j$. Using the law of cosines we get

$$\begin{aligned}
R_i^2 + R_j^2 - 2R_iR_j\cos(\overline{\varphi}_j - (\overline{\varphi}_i + k\beta_{ij})) &\geq (\rho_i + \rho_j)^2 \\
R_i^2 - 2R_iR_j + R_j^2 + 2R_iR_j - 2R_iR_j\cos(\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}) &\geq (\rho_i + \rho_j)^2 \\
(R_i - R_j)^2 + 2R_iR_j(1 - \cos(\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij})) &\geq (\rho_i + \rho_j)^2 \\
4R_iR_j\frac{1 - \cos\left(2\left[\frac{\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}}{2}\right]\right)}{2} &\geq (\rho_i + \rho_j)^2 - (R_i - R_j)^2 \\
\sin^2\left(\frac{\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}}{2}\right) &\geq \frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j} > 0. \qquad \text{(A.1)}
\end{aligned}$$

At this point, note that since

$$\begin{aligned}
k\beta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i &< (k+1)\beta_{ij} \\
0 \leq \overline{\varphi}_j - (\overline{\varphi}_i + k\beta_{ij}) &< \beta_{ij} \\
0 \leq \overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij} &< \frac{2\pi}{[n_i, n_j]} \\
0 \leq \frac{\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}}{2} &< \frac{\pi}{[n_i, n_j]} \leq \frac{\pi}{2}
\end{aligned}$$

inequality (A.1) yields

$$\begin{aligned}
\sin\left(\frac{\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}}{2}\right) &\geq \sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
\frac{\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}}{2} &\geq \arcsin\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij} &\geq 2\arcsin\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
\overline{\varphi}_j - \overline{\varphi}_i &\geq k\beta_{ij} + 2\arcsin\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
\overline{\varphi}_j - \overline{\varphi}_i &\geq k\beta_{ij} + \delta_{ij}. \qquad \text{(A.2)}
\end{aligned}$$

Similarly, we know that $P_{ij}(\overline{\varphi}_i, k+1) \cap P_j(\overline{\varphi}_j, 0) = \emptyset$ so their centers, $(R_i, \overline{\varphi}_i + (k+1)\beta_{ij})$ and

$(R_j, \overline{\varphi}_j)$, must also be distance at least $\rho_i + \rho_j$ apart. Thus

$$\begin{aligned}
R_i^2 + R_j^2 - 2R_iR_j\cos(\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j) &\geq (\rho_i + \rho_j)^2 \\
2R_iR_j - 2R_iR_j\cos(\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j) &\geq (\rho_i + \rho_j)^2 - (R_i^2 - 2R_iR_j + R_j^2) \\
2R_iR_j(1 - \cos(\overline{\varphi}_i - \overline{\varphi}_j + (k+1)\beta_{ij})) &\geq (\rho_i + \rho_j)^2 - (R_i - R_j)^2 + \\
4R_iR_j\frac{1 - \cos\left(2\left[\dfrac{\overline{\varphi}_i - \overline{\varphi}_j + (k+1)\beta_{ij}}{2}\right]\right)}{2} &\geq (\rho_i + \rho_j)^2 - (R_i - R_j)^2 \\
\sin^2\left(\frac{\overline{\varphi}_i - \overline{\varphi}_j + (k+1)\beta_{ij}}{2}\right) &\geq \frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j} > 0.
\end{aligned} \tag{A.3}$$

As before, note that since

$$\begin{aligned}
k\beta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i &< (k+1)\beta_{ij} \\
-\beta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i - (k+1)\beta_{ij} &< 0 \\
0 < \overline{\varphi}_i - \overline{\varphi}_j + (k+1)\beta_{ij} &\leq \beta_{ij} \\
0 < \overline{\varphi}_i - \overline{\varphi}_j + (k+1)\beta_{ij} &\leq \frac{2\pi}{[n_i, n_j]} \\
0 < \frac{\overline{\varphi}_i - \overline{\varphi}_j + (k+1)\beta_{ij}}{2} &\leq \frac{\pi}{[n_i, n_j]} \leq \frac{\pi}{2}
\end{aligned}$$

inequality (A.3) yields

$$\begin{aligned}
\sin\left(\frac{\overline{\varphi}_i - \overline{\varphi}_j + (k+1)\beta_{ij}}{2}\right) &\geq \sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
\frac{\overline{\varphi}_i - \overline{\varphi}_j + (k+1)\beta_{ij}}{2} &\geq \arcsin\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
\overline{\varphi}_i - \overline{\varphi}_j + (k+1)\beta_{ij} &\geq 2\arcsin\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
\overline{\varphi}_i - \overline{\varphi}_j &\geq -(k+1)\beta_{ij} + 2\arcsin\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
\overline{\varphi}_j - \overline{\varphi}_i &\leq (k+1)\beta_{ij} - 2\arcsin\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_iR_j}} \\
\overline{\varphi}_j - \overline{\varphi}_i &\leq (k+1)\beta_{ij} - \delta_{ij}.
\end{aligned} \tag{A.4}$$

Combining (A.2) and (A.4) gives

$$k\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - \delta_{ij}$$

as required.

To prove the uniqueness of k, consider the family of intervals

$$\big[\ell\beta_{ij} + \delta_{ij},\ (\ell+1)\beta_{ij} - \delta_{ij}\big], \quad \ell \in \mathbb{Z}.$$

We know that these intervals are all the same length, $\beta_{ij} - 2\delta_{ij}$, and that they must be non-empty since

$$\overline{\varphi}_j - \overline{\varphi}_i \in \big[k\beta_{ij} + \delta_{ij},\ (k+1)\beta_{ij} - \delta_{ij}\big].$$

We also know, assuming $\delta_{ij} > 0$, that the intervals are disjoint. Thus $\overline{\varphi}_j - \overline{\varphi}_i$ only lies in one of these intervals, that corresponding to k.

($\Leftarrow$)
For the remainder of this proof, let $\langle a, b\rangle$ denote the Euclidian distance between points a and b.

Let $\overline{\varphi}_i, \overline{\varphi}_j \in \mathbb{R}$ and $k \in \mathbb{Z}$ be given such that

$$k\beta_{ij} + \delta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} - \delta_{ij}.$$

First note that by definition $\delta_{ij} \geq 0$. The above then gives

$$\begin{aligned} k\beta_{ij} \leq \overline{\varphi}_j - \overline{\varphi}_i \leq (k+1)\beta_{ij} \\ \overline{\varphi}_i + k\beta_{ij} \leq \overline{\varphi}_j \leq \overline{\varphi}_i + (k+1)\beta_{ij}. \end{aligned} \tag{A.5}$$

Note that $(R_i, \overline{\varphi}_i + k\beta_{ij})$ and $(R_i, \overline{\varphi}_i + (k+1)\beta_{ij})$ are the centers of $P_{ij}(\overline{\varphi}_i, k)$ and $P_{ij}(\overline{\varphi}_i, k+1)$ respectively and $(R_j, \overline{\varphi}_j)$ is the center of $P_j(\overline{\varphi}_j, 0)$. Then (A.5) implies that the center of $P_j(\overline{\varphi}_j, 0)$ lies between the centers of sequential disks of $P_{ij}(\overline{\varphi}_i)$. Before we continue, we bound the distances

between these center points.

By our assumption,

$$\begin{aligned}
\overline{\varphi}_j - \overline{\varphi}_i &\geq k\beta_{ij} + \delta_{ij} \\
\overline{\varphi}_j - \overline{\varphi}_i &\geq k\beta_{ij} + 2\arcsin\sqrt{\frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j}} \\
\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij} &\geq 2\arcsin\sqrt{\frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j}} \\
\frac{\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}}{2} &\geq \arcsin\sqrt{\frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j}} \\
\sin\left(\frac{\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}}{2}\right) &\geq \sqrt{\frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j}}.
\end{aligned}$$

Squaring both sides gives

$$\begin{aligned}
\sin^2\left(\frac{\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}}{2}\right) &\geq \frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j} \\
4R_iR_j\frac{1-\cos\left(2\left[\frac{\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}}{2}\right]\right)}{2} &\geq (\rho_i+\rho_j)^2 - (R_i-R_j)^2 \\
(R_i-R_j)^2 + 2R_iR_j(1-\cos(\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij})) &\geq (\rho_i+\rho_j)^2 \\
R_i^2 - 2R_iR_j + R_j^2 + 2R_iR_j - 2R_iR_j\cos(\overline{\varphi}_j - \overline{\varphi}_i - k\beta_{ij}) &\geq (\rho_i+\rho_j)^2 \\
R_i^2 + R_j^2 - 2R_iR_j\cos(\overline{\varphi}_j - (\overline{\varphi}_i + k\beta_{ij})) &\geq (\rho_i+\rho_j)^2 \\
\left\langle (R_j, \overline{\varphi}_j), (R_i, \overline{\varphi}_i + k\beta_{ij}) \right\rangle &\geq \rho_i + \rho_j. \qquad \text{(A.6)}
\end{aligned}$$

Also by our initial assumption,

$$\begin{aligned}
\overline{\varphi}_j - \overline{\varphi}_i &\leq (k+1)\beta_{ij} - \delta_{ij} \\
\overline{\varphi}_j - \overline{\varphi}_i &\leq (k+1)\beta_{ij} - 2\arcsin\sqrt{\frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j}} \\
\overline{\varphi}_j - \overline{\varphi}_i - (k+1)\beta_{ij} &\leq -2\arcsin\sqrt{\frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j}} \\
\frac{\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j}{2} &\geq \arcsin\sqrt{\frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j}} \\
\sin\left(\frac{\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j}{2}\right) &\geq \sqrt{\frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j}}.
\end{aligned}$$

Squaring both sides gives

$$\begin{aligned}
\sin^2\left(\frac{\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j}{2}\right) &\geq \frac{(\rho_i+\rho_j)^2 - (R_i-R_j)^2}{4R_iR_j} \\
4R_iR_j\frac{1-\cos\left(2\left[\frac{\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j}{2}\right]\right)}{2} &\geq (\rho_i+\rho_j)^2 - (R_i-R_j)^2 \\
(R_i-R_j)^2 + 2R_iR_j(1-\cos(\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j)) &\geq (\rho_i+\rho_j)^2 \\
R_i^2 - 2R_iR_j + R_j^2 + 2R_iR_j - 2R_iR_j\cos(\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j) &\geq (\rho_i+\rho_j)^2 \\
R_i^2 + R_j^2 - 2R_iR_j\cos(\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j)) &\geq (\rho_i+\rho_j)^2 \\
\left\langle (R_i, \overline{\varphi}_i + (k+1)\beta_{ij}), (R_j, \overline{\varphi}_j)\right\rangle &\geq \rho_i + \rho_j. \qquad \text{(A.7)}
\end{aligned}$$

Now, let $(\overline{r}, \overline{\theta}) \in P_{ij}(\overline{\varphi}_i)$ be given. Without loss of generality, assume that $\overline{\varphi}_j - \pi \leq \overline{\theta} < \overline{\varphi}_j + \pi$. We will now prove that the distance between the center of $P_j(\overline{\varphi}_j, 0)$ and $(\overline{r}, \overline{\theta})$ is greater than ρ_j. This will show that $P_j(\overline{\varphi}_j, 0) \cap P_{ij}(\overline{\varphi}_i) = \emptyset$. To do this, we need to break into cases depending on the value of $\overline{\theta}$. In an attempt to clarify these cases, Figure A.1 shows the partition of $P_{ij}(\overline{\varphi}_i)$.

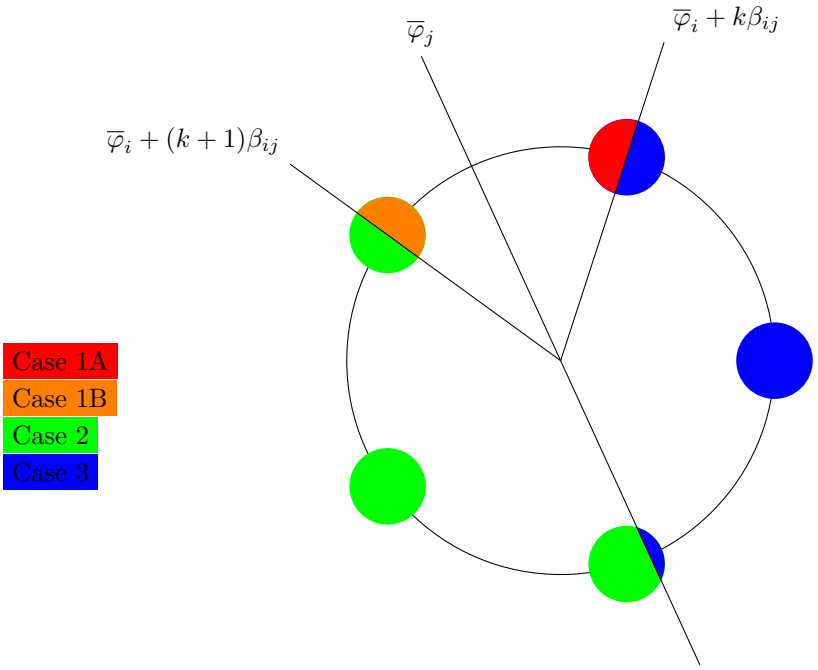


Figure A.1: Breakdown of possible $(\overline{r}, \overline{\theta}) \in P_{ij}(\overline{\varphi}_i)$ into cases

Case 1 ($\overline{\varphi}_i + k\beta_{ij} \leq \overline{\theta} \leq \overline{\varphi}_i + (k+1)\beta_{ij}$):

Note that, since $P_{ij}(\overline{\varphi}_i, k)$ and $P_{ij}(\overline{\varphi}_i, k+1)$ are sequential disks of P_{ij}, if

$$\overline{\varphi}_i + k\beta_{ij} \leq \overline{\theta} \leq \overline{\varphi}_i + (k+1)\beta_{ij}$$

then $(\overline{r}, \overline{\theta})$ must be in one of these two disks.

Case 1A ($(\overline{r}, \overline{\theta}) \in P_{ij}(\overline{\varphi}_i, k)$):

Recall from (A.6) that the distance between the centers of disks $P_{ij}(\overline{\varphi}_i, k)$ and $P_j(\overline{\varphi}_j, 0)$ is

$$\Big\langle (R_j, \overline{\varphi}_j), (R_i, \overline{\varphi}_i + k\beta_{ij}) \Big\rangle \geq \rho_i + \rho_j.$$

Also, since $(\overline{r}, \overline{\theta}) \in P_{ij}(\overline{\varphi}_i, k)$,

$$\Big\langle (\overline{r}, \overline{\theta}), (R_i, \overline{\varphi}_i + k\beta_{ij}) \Big\rangle < \rho_i.$$

Thus, using the triangle inequality, we get

$$\Big\langle (R_j, \overline{\varphi}_j), (\overline{r}, \overline{\theta}) \Big\rangle \geq \Big\langle (R_j, \overline{\varphi}_j), (R_i, \overline{\varphi}_i + k\beta_{ij}) \Big\rangle - \Big\langle (\overline{r}, \overline{\theta}), (R_i, \overline{\varphi}_i + k\beta_{ij}) \Big\rangle > \rho_j.$$

Thus $(\overline{r}, \overline{\theta}) \notin P_j(\overline{\varphi}_j, 0)$.

Case 1B ($(\overline{r}, \overline{\theta}) \in P_{ij}(\overline{\varphi}_i, k+1)$):

Recall from (A.7) that the distance between the centers of disks $P_{ij}(\overline{\varphi}_i, k+1)$ and $P_j(\overline{\varphi}_j, 0)$ is

$$\Big\langle (R_i, \overline{\varphi}_i + (k+1)\beta_{ij}), (R_j, \overline{\varphi}_j) \Big\rangle \geq \rho_i + \rho_j.$$

Also, since $(\overline{r}, \overline{\theta}) \in P_{ij}(\overline{\varphi}_i, k+1)$,

$$\Big\langle (\overline{r}, \overline{\theta}), (R_i, \overline{\varphi}_i + (k+1)\beta_{ij}) \Big\rangle < \rho_i.$$

Thus, using the triangle inequality, we get

$$\Big\langle (R_j, \overline{\varphi}_j), (\overline{r}, \overline{\theta}) \Big\rangle \geq \Big\langle (R_j, \overline{\varphi}_j), (R_i, \overline{\varphi}_i + (k+1)\beta_{ij}) \Big\rangle - \Big\langle (\overline{r}, \overline{\theta}), (R_i, \overline{\varphi}_i + (k+1)\beta_{ij}) \Big\rangle > \rho_j.$$

Thus $(\overline{r}, \overline{\theta}) \notin P_j(\overline{\varphi}_j, 0)$.

Case 2 ($\overline{\theta} < \overline{\varphi}_i + k\beta_{ij}$):

Combining our assumptions and $\delta_{ij} \geq 0$, we get

$$\begin{gathered}
\overline{\varphi}_j - \pi \leq \overline{\theta} < \overline{\varphi}_i + k\beta_{ij} \leq \overline{\varphi}_j \\
-\pi \leq \overline{\theta} - \overline{\varphi}_j < \overline{\varphi}_i + k\beta_{ij} - \overline{\varphi}_j \leq 0 \\
0 \leq \overline{\varphi}_j - (\overline{\varphi}_i + k\beta_{ij}) < \overline{\varphi}_j - \overline{\theta} \leq \pi \\
\cos\big(\overline{\varphi}_j - (\overline{\varphi}_i + k\beta_{ij})\big) > \cos(\overline{\varphi}_j - \overline{\theta})
\end{gathered}$$

since cosine is strictly decreasing on the interval $(0, \pi)$.

Now consider the point $(R_i, \overline{\theta})$.

$$\begin{aligned}\Big\langle (R_j, \overline{\varphi}_j), (R_i, \overline{\theta}) \Big\rangle^2 &= R_i^2 + R_j^2 - 2R_iR_j\cos(\overline{\varphi}_j - \overline{\theta}) \\ &> R_i^2 + R_j^2 - 2R_iR_j\cos\big(\overline{\varphi}_j - (\overline{\varphi}_i + k\beta_{ij})\big) \\ &= \Big\langle (R_j, \overline{\varphi}_j), (R_i, \overline{\varphi}_i + k\beta_{ij}) \Big\rangle^2 \\ &\geq (\rho_i + \rho_j)^2\end{aligned}$$

by (A.6). Finally,

$$\begin{aligned}\Big\langle (\overline{r}, \overline{\theta}), (R_i, \overline{\theta}) \Big\rangle^2 &= \overline{r}^2 + R_i^2 - 2\overline{r}R_i\cos(\overline{\theta} - \overline{\theta}) \\ &= \overline{r}^2 + R_i^2 - 2\overline{r}R_i \\ &= (R_i - \overline{r})^2 \\ &< \rho_i^2\end{aligned}$$

by Property 2.1.3(d). Thus, once again by the triangle inequality,

$$\Big\langle (R_j, \overline{\varphi}_j), (\overline{r}, \overline{\theta}) \Big\rangle \geq \Big\langle (R_j, \overline{\varphi}_j), (R_i, \overline{\theta}) \Big\rangle - \Big\langle (\overline{r}, \overline{\theta}), (R_i, \overline{\theta}) \Big\rangle > \rho_j.$$

Case 3 ($\overline{\theta} > \overline{\varphi}_i + (k+1)\beta_{ij}$):

Combining our assumptions and $\delta_{ij} \geq 0$, we get

$$\begin{aligned}\overline{\varphi}_j \leq \overline{\varphi}_i + (k+1)\beta_{ij} < \overline{\theta} < \overline{\varphi}_j + \pi \\ 0 \leq \overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j < \overline{\theta} - \overline{\varphi}_j \leq \pi \\ \cos\big(\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j\big) > \cos(\overline{\theta} - \overline{\varphi}_j)\end{aligned}$$

since cosine is strictly decreasing on the interval $(0, \pi)$.

Again, consider the point $(R_i, \overline{\theta})$.

$$\begin{aligned}
\Big\langle (R_j, \overline{\varphi}_j), (R_i, \overline{\theta}) \Big\rangle^2 &= R_i^2 + R_j^2 - 2R_iR_j\cos(\overline{\theta} - \overline{\varphi}_j) \\
&> R_i^2 + R_j^2 - 2R_iR_j\cos\big(\overline{\varphi}_i + (k+1)\beta_{ij} - \overline{\varphi}_j\big) \\
&= \Big\langle (R_j, \overline{\varphi}_j), \big(R_i, \overline{\varphi}_i + (k+1)\beta_{ij}\big) \Big\rangle^2 \\
&\geq (\rho_i + \rho_j)^2
\end{aligned}$$

by (A.7). Finally,

$$\begin{aligned}
\Big\langle (\overline{r}, \overline{\theta}), (R_i, \overline{\theta}) \Big\rangle^2 &= \overline{r}^2 + R_i^2 - 2\overline{r}R_i\cos(\overline{\theta} - \overline{\theta}) \\
&= \overline{r}^2 + R_i^2 - 2\overline{r}R_i \\
&= (R_i - \overline{r})^2 \\
&< \rho_i^2
\end{aligned}$$

by Property 2.1.3(d). Thus, by the triangle inequality,

$$\Big\langle (R_j, \overline{\varphi}_j), (\overline{r}, \overline{\theta}) \Big\rangle \geq \Big\langle (R_j, \overline{\varphi}_j), (R_i, \overline{\theta}) \Big\rangle - \Big\langle (\overline{r}, \overline{\theta}), (R_i, \overline{\theta}) \Big\rangle > \rho_j.$$

In all cases $(\overline{r}, \overline{\theta}) \notin P_j(\overline{\varphi}_j, 0)$. This implies that $P_{ij}(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j, 0) = \emptyset$ which, by Lemmas 2.2.1 and 2.2.2, gives us $P_i(\overline{\varphi}_i) \cap P_j(\overline{\varphi}_j) = \emptyset$ as required.

■

Bibliography

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Upper Saddle River, NJ, 1993.

[2] S. Anily, C. A. Glass, and R. Hassin. The scheduling of maintenance service. *Discrete Applied Mathematics*, 82(1-3):27–42, March 1998.

[3] A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieber. Minimizing service and operation costs of periodic scheduling. *Mathematics of Operations Research*, 27(3):518–544, August 2002.

[4] A. Bar-Noy, V. Dreizin, and B. Patt-Shamir. Efficient algorithms for periodic scheduling. *Computer Networks*, 45(2):155–173, June 2004.

[5] S. K. Baruah, R. R. Howell, and L. E. Rosier. Feasibility problems for recurring tasks on one processor. *Theoretical Computer Science*, 118(1):3–20, September 1993.

[6] A. R. Brown. *Optimum Packing and Depletion*, Volume 14 of *Computer Monographs*. Macdonald and Co., New York, NY, 1971.

[7] R. E. Burkhard. Optimal schedules for periodically recurring events. *Discrete Applied Mathematics*, 15(2-3):167–180, November 1986.

[8] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, Chapter 2, pages 46–93. PWS Publishing Co., Boston, MA, USA, 1997.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, Second edition, 2007.

[10] S. K. Dhall and C. L. Liu. On a real-time scheduling problem. *Operations Research*, 26(1):127–140, January - February 1978.

[11] F. Eisenbrand, K. Kesavan, R. Mattikalli, M. Niemeier, A. Nordsieck, M. Skutella, J. Verschae, and A. Wiese. Solving an avionics real-time scheduling problem by advanced IP-methods. In M. de Berg and U. Meyer, editors, *Algorithms – ESA 2010*, Volume 6346 of *Lecture Notes in Computer Science*, pages 11–22. Springer Berlin / Heidelberg, 2010.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, 1979.

[13] A. Grigoriev, J. van de Klundert, and F. C. R. Spieksma. Modeling and solving the periodic maintenance problem. *European Journal of Operational Research*, 172(3):783–797, August 2006.

[14] Gurobi Optimization, Inc., Houston, TX. *Gurobi Optimizer Version 4.5.1*, April 2011.

[15] D. S. Hochbaum, N. Megiddo, J. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Mathematial Programming*, 62(1-3):69–83, 1993.

[16] K. Ireland and M. Rosen. *A Classical Introduction to Modern Number Theory*. Springer Science+Business Media, New York, NY, 1990.

[17] J. Korst, E. Aarts, and J. K. Lenstra. Scheduling periodic tasks. *INFORMS Journal on Computing*, 8(4):428–435, Fall 1996.

[18] J. Korst, E. Aarts, J. K. Lenstra, and J. Wessels. Periodic multiprocessor scheduling. In *Proceedings of the Conference on Parallel Architectures and Languages Europe, PARLE '91*, Volume 505 of *Lecture Notes in Computer Science*, pages 166–178. Springer, Berlin, 1991.

[19] J. Korst, E. Aarts, J. K. Lenstra, and J. Wessels. Periodic assignment and graph colouring. *Discrete Applied Mathematics*, 51(3):291–305, July 1994.

[20] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.

[21] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, January 1998.

[22] J. B. Orlin. Minimizing the number of vehicles to meet a fixed periodic schedule: An application of periodic posets. *Operations Research*, 30(4):760–776, July - August 1982.

[23] K. S. Park and D. K. Yun. Optimal scheduling of periodic activities. *Operations Research*, 33(3):690–695, May - June 1985.

[24] B. Paynter. An optimization approach to constructing wheel lug hole templates. Masters project, Department of Mathematical Sciences, Clemson University, Clemson, SC, May 2008.

[25] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6(1):64–94, 1962.

[26] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, November 1989.

[27] A. Vince. Scheduling periodic events. *Discrete Applied Mathematics*, 25(3):229–310, November 1989.

[28] W. D. Wei and C. L. Liu. On a periodic maintenance problem. *Operations Research Letters*, 2(2):90–93, June 1983.

[29] L. A. Wolsey. *Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, New York, NY, 1998.