2011

Hamilton-Waterloo problem with triangle and C9 factors

David C. Kamin
Michigan Technological University

THE HAMILTON-WATERLOO PROBLEM WITH TRIANGLE AND C_9 FACTORS

By

David C. Kamin

A THESIS

Submitted in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

Mathematical Sciences

MICHIGAN TECHNOLOGICAL UNIVERSITY

2011

This thesis, "The Hamilton-Waterloo Problem with Triangle and C_9 factors", is hereby approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE IN MATHEMATICAL SCIENCES.

Department of Mathematical Sciences

Signatures:

Thesis Advisor _____
Dr. Melissa Keranen

Co-Advisor _____
Dr. Sibel Özkan

Department Chair _____
Dr. Mark Gockenbach

Date _____

Contents

List of Figures

Acknowledgements

This work is dedicated first and foremost to my parents. If not for their unconditional love, support, and advice these past 25 years, I would not have had the courage to be who or where I am today. Thank you.

Oma - I know you would be proud of me if you could see this. I know you would be proud of me no matter what. And that means a lot.

Samantha Ball - for your love, your cooking, your smile, and your tolerance of my near-constant nonsense - Thank you. I could not have done this without your support.

Melissa Keranen and Sibel Özkan - this is a result of your mathematical expertise and guidance, none of which went unappreciated. Thank you.

John Asplund - I *may* have been able to do this without your help, but I absolutely would not have been able to finish it when I did, or enjoy it even half as much. For your advice, your encouragement, your help, and most of all, your friendship - Thank you.

Steven Lewis - for all the times you set aside your work to help with mine - Thank you. Your editing expertise is much appreciated.

William Laffin - it seems like every time I spoke to you about my project, you would drop whatever you were working on to listen, help me understand, and help me program. I appreciate it.

Melanie Laffin - it was good to have someone fighting through the writing process and stylistic procedures alongside me.

Richard Fears, David Clark, Josh Ruark, Michael Mission, James Wright, and the rest of the the graduate students in the math department - it has been a pleasure working alongside you.

Don Kreher - I am grateful for your interest in, ideas about, and perspective on the K_{18} case.

Mark Gockenbach, Thomas Drummer, Ann Humes, and the rest of the faculty in the math department - it has been a pleasure working under you.

Margaret Perander, Tori Connors, and Jeanne Meyers - I believe you are the reason that the math department is able to accomplish anything on a day-to-day basis. Thank you.

Abstract

The Hamilton-Waterloo problem and its spouse-avoiding variant for uniform cycle sizes asks if K_v, where v is odd (or $K_v - F$, if v is even), can be decomposed into 2-factors in which each factor is made either entirely of m-cycles or entirely of n-cycles. This thesis examines the case in which r of the factors are made up of cycles of length 3 and s of the factors are made up of cycles of length 9, for any r and s. We also discuss a constructive solution to the general (m,n) case which fixes r and s.

Chapter 1

Introduction

1.1 Graph Decomposition

A *graph* is a collection of points, pairs of which may or may not be connected by edges. They are very useful modeling tools for discrete situations. If one lives in a very snowy climate, for example, one might be concerned with the efficiency of the road commission's plowing efforts after a storm. Imagining each plow traveling down each road might seem like a daunting task, but we can model the path of each plow using a graph. We will use road intersections as points and road segments as edges. The path each plow takes must be connected, each path must start and end at the point representing the road commission, and the union of all the paths must cover each edge at least once.

With this model, the problem becomes a bit easier. Given a few different solutions to this problem, one can look at how much overlap different paths have (multiple trucks plowing the same section of road) and how long each path is to determine which solution is more efficient than the others.

In this problem, the path a truck takes may cross itself, and aside from efficiency reasons, there is no reason multiple trucks (paths) cannot use the same streets (edges). That makes this problem, as stated, not a proper cycle decomposition, but it is related.

A *cycle* is a connected graph such that each vertex is incident to exactly two edges. Looking at what cycles are present within larger graphs can tell us a lot about those graphs. For example, consider G_1 and G_2 as given in Figure 1.1 and Figure 1.2. Are they isomorphic?

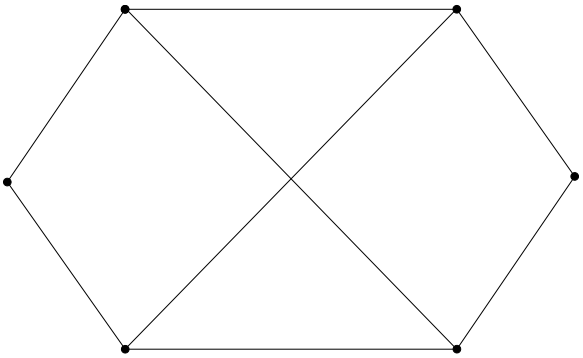

Figure 1.1: H_1, a graph on six vertices

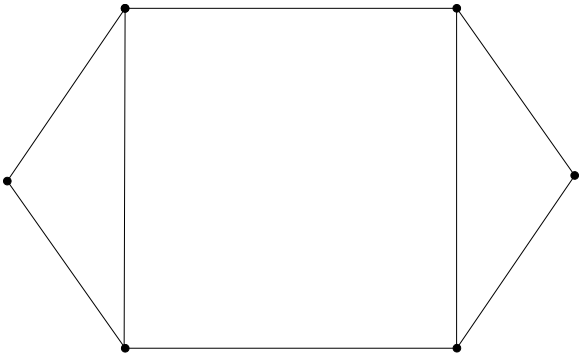

Figure 1.2: H_2, a graph on six vertices

We can see at a glance that the two graphs have the same number of vertices and edges, so using that to quickly rule out an isomorphism will not work. The next thing to check is whether or not they contain the same cycles. H_1 contains a cycle of length 6 and five cycles of length 4. H_2, contains a cycle of length 6, two cycles of length 5, a cycle of length 4,

and two cycles of length 3. Because these two lists are not identical, H_1 and H_2 cannot be isomorphic.

Again, this is not a proper cycle decomposition. A *cycle decomposition* is a partition of the edge-set of a graph into cycles. Each edge must be used exactly once. In the snowplow problem, the paths might not have been cycles. In the isomorphism problem, each edge was used multiple times (in fact, neither H_1 or H_2 have any cycle decompositions at all).

A proper cycle decomposition problem might look something like this: When does K_v, the complete graph on v vertices, have a cycle decomposition into cycles of length v? This problem is known as the Hamiltonian decomposition of the complete graph, and it is actually a special case of the Oberwolfach Problem.

1.2 The Oberwolfach Problem

Before we begin to understand the Hamilton-Waterloo problem, we must first understand the well known Oberwolfach problem.

The Oberwolfach problem was proposed by Ringel in 1967 at a conference in its namesake: the city of Oberwolfach, Germany. The problem involves trying to seat v conference attendees at t round tables over $\frac{v-1}{2}$ nights such that each attendee sits next to each other attendee exactly once.

As originally stated, it is clear that v must be odd, and the total number of seats at the t round tables must be v (so that everyone has a neighbor for each meal), but it is not clear how large each table is. A conference of 25 people might have one large table that seats 8, two smaller tables that seat 5 each, a table for 4 and a table for 3. When we begin to imagine all the different possibilities, the difficulty of the problem becomes apparent.

Even more possibilities arise when we consider an extension of the Oberwolfach Problem called the *spouse avoiding variant*. This version of the problem allows for an even

number of attendees and seats them over $\frac{v-2}{2}$ nights, with the stipulation that each attendee sits next to each other attendee - except his or her spouse - exactly once. This variant is often lumped in with the original Oberwolfach Problem to remove the "v must be odd" restriction. This combination problem is still called the Oberwolfach Problem, and it is assumed that spouses will be avoided if v is even. For the rest of this thesis, "The Oberwolfach Problem" will refer to both the original problem and the spouse avoiding variant.

The difficulty in solving the Oberwolfach Problem is handled by restricting the number of different table sizes. The above example with 25 people uses four different table sizes (8, 5, 4, and 3), making it fairly complicated. The simplest cases have only one table size, or in other words, the table size is constant. To seat 25 people around tables of a constant size, there are only two options: one table of 25 people or five tables of 5 people each. Already we can see that the problem has been simplified considerably.

As we will see in Section 1.6, the Oberwolfach Problem has been completely solved for constant table sizes. The Oberwolfach Problem, however, is not the focus of this thesis.

1.3 The Hamilton-Waterloo Problem

Now that we have a thorough understanding of the Oberwolfach Problem, we can talk about its most popular variant: the Hamilton-Waterloo problem. The Hamilton-Waterloo problem takes the same idea of seating attendees, but it splits the conference between two dining halls, each with a different set of table sizes. There are still v attendees, and the conference still lasts for $\frac{v-1}{2}$ nights, but each person spends the same r of those nights at a venue in Hamilton, and the other $s = \frac{v-1}{2} - r$ nights at a venue in Waterloo. The goal is still to devise a seating arrangement such that each attendee sits beside each other attendee exactly once throughout the conference.

Much of what was true about the Oberwolfach Problem in Section 1.2 is also true of the Hamilton-Waterloo problem. For example, as stated, v must be odd, but there is again a spouse avoiding variant that allows for an even number of attendees. Again, in this case, the conference will last for $\frac{v-2}{2}$ nights, and each attendee must sit beside each other attendee except his or her spouse. As with the Oberwolfach Problem, we will lump the original Hamilton-Waterloo problem together with its spouse avoiding variant throughout the rest of this thesis. This will allow for any number of attendees, and it will be assumed that spouses will be avoided if v is even.

The other major similarity the Hamilton-Waterloo problem has with the Oberwolfach Problem is that, as stated, there is no guarantee that the tables in Hamilton or in Waterloo are of uniform size. And again, to simplify the problem, we assume that each table in Hamilton is uniform of size m, and each table in Waterloo is uniform of size n.

At a glance, it may not be clear that the Hamilton-Waterloo problem is different from the Oberwolfach Problem on two different table sizes, but the differences are vast. In the Oberwolfach Problem with two different table sizes, the tables of different sizes are filled simultaneously. A conference of 9 people could be seated between a table of size 4 and a table of size 5 each night. In the Hamilton-Waterloo problem, all the attendees must be seated at tables of the same size on any given night. Perhaps one night, they all sit at tables of size 5, and the next, they all sit at tables of size 4. In this case, there must be at least 20 attendees.

To say that a particular case of m and n is solved one should prove that a successful seating arrangement is either possible or impossible for every applicable number of attendees v, nights in Hamilton r, and nights in Waterloo s. This thesis represents the author's attempts to do that for $m = 3$ and $n = 9$. But first, we need to examine the tools we have and draw some connections between problems of seating arrangements and problems of mathematics. We begin with Graph Theory.

1.4 Graph Theory

Simply speaking, a graph is a collection of points, each pair of which may or may not be connected by an edge. Mathematically, we say that a *graph* G is a pair (V,E), where V is a set of v points, called*vertices*, and E is a set of distinct pairs of distinct points, called*edges*.

Graphs are particularly effective modeling tools for a variety of situations. The railways of a large, complicated city like Tokyo can be modeled with graphs, using stations as vertices and train lines as edges. This can be used to make maps and help people find their way from point A to point B.

For most of this thesis, we will be dealing with one of two very specific graphs: K_v and $K_{(h:u)}$. The*complete graph* on v vertices, K_v, is a graph in which each of the $\binom{v}{2}$ pairs of vertices are connected by edges. If each node represents a conference attendee and an edge between nodes represents "These two people have been neighbors at a meal," the complete graph can be used to model the result "each attendee has been each other attendee's neighbor exactly once." An *equipartite graph* is a graph whose vertex set can be partitioned into u subsets of size h such that no two vertices from the same subset are connected by an edge. The complete equipartite graph with u subsets of size h is denoted $K_{(h:u)}$, and it contains every edge between vertices of different subsets. To use these graphs, we will need the following tools.

A*subgraph* G' of a graph G is a pair (V',E') with $V' \subseteq V$, $E' \subseteq E$. An *induced subgraph* G' of a graph G is a subgraph such that E' contains all the edges in E between the points of V'. A *spanning subgraph* G' of a graph G is a subgraph such that $V' = V$. For example, $K_{(4:5)}$ is a spanning subgraph of K_{20}. If you took one point from each of the 5 partitions of $K_{(4:5)}$, the resulting induced subgraph would be K_5.

The *degree* of a vertex is the number of edges incident to it. A graph G is said to be *regular of degree* k (sometimes k-regular) if and only if the degree of each vertex is k. K_v is regular of degree $v-1$, $K_{(h:u)}$ is regular of degree $h(u-1)$.

A graph is said to be *connected* if, starting from any $\alpha \in V$, you can reach any $\beta \in V$ by traversing edges. A *cycle* is a connected 2-regular graph. A cycle on n points is called an *n-cycle*, and it is often written as C_n. Cycles of length 3 are sometimes called *triangles* for obvious reasons.

If two graphs, G and H, have the same vertex set V, we can talk about their direct sum by taking the union of their edge-sets. The *direct sum* of two graphs $G = (V, E_1)$ and $H = (V, E_2)$ is $G \oplus H = (V, E_1 \cup E_2)$.

A *k-factor* of a graph G is a k-regular spanning subgraph. A 1-factor, sometimes called a *perfect matching*, is often simply denoted by F. A *k-factorization* of a graph G is a collection of k-factors such that each edge of G is used exactly once. Figure 1.3 is an example of a 2-factor of K_9, and Figure 1.4 gives a full 2-factorization of K_9 into 3-cycles. Note that each 2-factor of a graph uses 2 edges incident to each vertex, so for a graph to have a 2-factorization, it must be regular of even degree. When v is odd, K_v satisfies this property, and when v is even, $K_v - F$ satisfies this property.

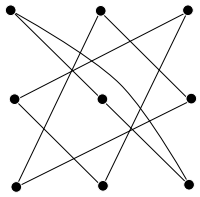

Figure 1.3: A 2-factor of K_9

Throughout this thesis, the word *factor* is assumed to be a 2-factor unless otherwise stated. Similarly, a *triangle factor* is a factor of 3-cycles and a *C_9-factor* is a factor of 9-cycles.

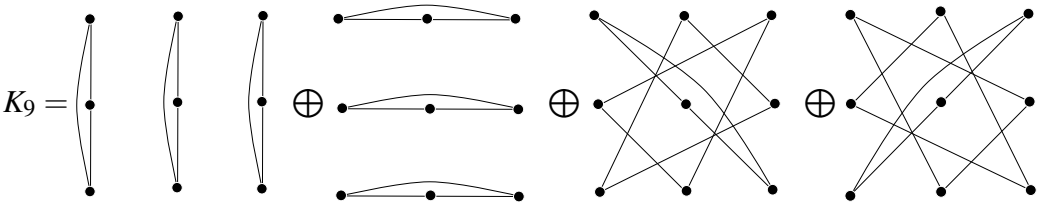

Figure 1.4: A 2-factorization of K_9

Now it is time we examined the connection between the Hamilton-Waterloo problem of seating arrangements and graph theory. For a more in-depth look at Graph Theory, see [6].

1.5 The Mathematical Connection

As was hinted about earlier, "each of v attendees sits next to each other attendee exactly once" can be modeled by the complete graph on v vertices, where each vertex represents a person and each edge between two vertices represents the two people having sat next to each other one night. In this model, a round table can be represented by a cycle, which is by definition 2-regular, and a dining hall full of v people at round tables can be represented by a 2-factor of K_v. If each of the tables has size m (like in Hamilton), then the 2-factor would be made entirely out of m-cycles. If each of the tables has size n (like in Waterloo), then the 2-factor would be made entirely out of n-cycles. A collection of nightly seating arrangements that satisfy the property that each person sits next to each other person exactly once over the course of the conference can be represented by a 2-factorization of K_v, where each factor is again made entirely out of m-cycles or entirely out of n-cycles, depending on which city the people dined in that night. To account for spousal avoidance, we simply have to remove a 1-factor, F, from K_v. The edges of F represent marriage, and if two people are married, then they do not wish to sit next to each other. Therefore, a 2-factorization of $K_v - F$ accounts for each edge, either through seating arrangements or marriage.

To recap: the Hamilton-Waterloo problem of seating v conference attendees for r nights in Hamilton, with tables that each seat m people, and s nights in Waterloo, with tables that seat n people, where $r+s=\frac{v-1}{2}$ (or $r+s=\frac{v-2}{2}$, if v is even) nights such that each attendee sits beside each other attendee (except possibly his or her spouse) exactly once throughout the conference is mathematically equivalent to decomposing K_v (or K_v-F, is v is even) into 2-factors, r of which are made entirely out of m-cycles and s of which are entirely out of n-cycles. When such a factorization exists, it is called a *Uniformly Resolvable Decomposition*, abbreviated (m,n)–URD$(v;r,s)$.

Similarly, the Oberwolfach Problem can be modeled using graph theory. The Oberwolfach Problem of seating $v=nt$ attendees at t round tables, each of size n, over $\frac{v-1}{2}$ (or $\frac{v-2}{2}$, if v is even) nights is equivalent to finding a decomposition of K_v (or K_v-F, if v is even) into n-cycle factors. This problem will be referred to as OP(n;t).

1.6 Previous Results

The Oberwolfach Problem for constant cycle length (constant table size) was completely solved between Alspach and Haggkvist; Alspach, Schellenberg, Stinson, and Wagner; and Hoffman and Schellenberg. The three papers were published in 1985 [2], 1989 [3], and 1991 [9], respectively. The results were impressive: Given any number v of guests and any constant table size m that divides v, a solution exists, with two exceptions. These results are summarized in Theorem 1.1.

Theorem 1.1. *[2, 3, 9] OP$(n;t)$ has a solution for all n,t except $(n;t)\in\{(3;2),(3;4)\}$.*

Another key result, provided in 2003 by Liu [13], solved a variation of the Oberwolfach Problem, one that seeks to decompose $K_{(h:u)}$ instead of K_v. This is the logical extenion of the spouse avoiding variant - instead of everyone having just one person they want to avoid, it assumes that each person is part of a group of h people that they want to avoid. This

problem was also solved completely. That is to say, with finitely many exceptions, $K_{(h:u)}$ has a C_m-factorization.

Theorem 1.2. *[13] For $m \geq 3$ and $u \geq 2$, $K_{(h:u)}$ has a C_m-factorization if and only if hu is divisible by m, $h(u-1)$ is even, m is even if $u = 2$, and $(h,u,m) \neq (2,3,3)$, $(6,3,3)$, $(2,6,3)$, $(6,2,6)$.*

The first breakthrough results on the Hamilton-Waterloo problem came in 2002. Adams, Billington, Bryant, and El-Zanati solved the Hamilton-Waterloo problem for a host of pairs (m,n) using base cases and recursive constructions. They covered the following cases: $(m,n) \in \{(4,6),(4,8),(4,16),(8,16),(3,5),(3,15),(5,15)\}$. Furthermore, they provided examples of solutions to all Hamilton-Waterloo problems on less than 17 vertices [1].

One year later, Horak, Nedela, and Rosa mostly solved the case of Triangle factors and Hamilton Cycle factors on an odd number of vertices [10]. A *Hamilton cycle* is a cycle on v points. Some of the exceptions for this case were later resolved by Dinitz and Ling in 2009 [7, 12].

Later in 2008, Fu and Huang settled all cases where $m = 4$ and n is even, and also settled all cases where $n = 2m$ [8]. They used an inventive direct construction.

Save for a few exceptions, Danziger, Quattrocchi, and Stevens solved Hamilton-Waterloo problem for 3-cycles and 4-cycles, marking the first solution with cycle sizes of mixed parity. These results were published in 2009 [5].

The latest results of which the author is aware as of this writing also came in 2010, when Keranen and Özkan settled the case of 4-cycles and a single factor of n-cycles [11].

The necessary conditions for the existence of a (m,n)–$\mathsf{URD}(v;r,s)$ were first put forth in 2002 by Adams, Billington, Bryant, and El-Zanati [1]. Clearly, $r+s = \frac{v-1}{2}$ (or $r+s = \frac{v-2}{2}$, for even v) and both m and n must divide v. These conditions are summarized below.

Theorem 1.3. *[1] The necessary conditions for the existence of an* (m,n)–$\mathsf{URD}(v;r,s)$ *are*

1. *If* v *is odd,* $r+s=\frac{v-1}{2}$,
2. *If* v *is even,* $r+s=\frac{v-2}{2}$,
3. *If* $r>0$, $m|v$,
4. *If* $s>0$, $n|v$.

To solve the $(m,n)=(3,9)$ case, we need one more basic tool: designs.

1.7 Design Theory

Designs, like graphs, are powerful tools in discrete mathematics. This thesis relies heavily on the resolvability of some designs into parallel classes, as explained below.

A *group divisible design* (k,λ)–$\mathsf{GDD}(h^u)$ is a triple $(\mathcal{V},\mathcal{G},\mathcal{B})$ where $\mathcal{V}$ is a finite set of size $v=hu$, $\mathcal{G}$ is a partition of $\mathcal{V}$ into *u groups* each containing h elements, $\mathcal{B}$ is a collection of k element subsets of $\mathcal{V}$ called *blocks* that satisfy:

- If $B\in\mathcal{B}$, then $|B|=k$.
- If a pair of elements from $\mathcal{V}$ appear in the same group, then the pair cannot be in any block.
- Two points that are not in the same group, called a *transverse pair*, appear in exactly λ blocks.
- $|\mathcal{G}|>1$.

A *resolvable GDD* (RGDD) has the additional condition that the blocks can be partitioned into parallel classes such that each element of $\mathcal{V}$ appears exactly once in each

parallel class. A *uniform* RGDD is one in which $\lambda = 1$. We denote such an RGDD as a k–RGDD(h^u). For the purposes of this thesis, we will only talk about *uniform* RGDDs.

The necessary conditions for the existence of an RGDD are well established and given below.

Theorem 1.4. *[4] The necessary conditions for the existence of a k–RGDD(h^u) are:*

1. $u \geq k$,
2. $hu \equiv 0 \pmod{k}$,
3. $h(u-1) \equiv 0 \pmod{k-1}$.

Sufficient conditions for the existence of (k,λ)–RGDD(h^u)s have been discovered for $k = 2,3,4$ and $\lambda = 1$ except in a finite number of cases.

Theorem 1.5. *[14] A $(3,\lambda)$–RGDD(h^u) exists if and only if $u \geq 3$, $\lambda h(u-1)$ is even, $hu \equiv 0 \pmod{3}$, and $(\lambda,h,u) \notin \{(1,2,6),(1,6,3)\} \bigcup \{(2j+1,2,3),(4j+2,1,6) : j \geq 0\}$.*

In particular, we have that a 3–RGDD(3^u) exists for all odd $u \geq 3$ and a 3–RGDD(6^u) exists for all $u \geq 4$.

For a more in-depth look at Design Theory, see [15].

Chapter 2

Generating URDs for fixed r, s

The first original theorem presented here is a direct result of applying Theorem 1.1 and Theorem 1.2 together. Given m and n, it can generate individual solutions to the Hamilton-Waterloo problem for a specific r and s.

Theorem 2.1. *Let $v = hu, h, u \geq 2$ be given and choose m and n such that the following conditions hold.*

1. $n|h$,
2. $(n,h) \neq (3,6),(3,12)$,
3. $m|v$,
4. $h(u-1)$ *is even,*
5. m *is even if* $u = 2$,
6. $(h,u,m) \neq (2,3,3),(6,3,3),(2,6,3),(6,2,6)$.

Then there exists a (m,n)–$\mathsf{URD}(v;r,s)$, *where* $r = \frac{h(u-1)}{2}$ *and* $s = \frac{h-1}{2}$ *if* v *is odd or* $s = \frac{h-2}{2}$ *is* v *is even.*

Proof: We begin by partitioning the vertex set of K_v into u groups of size h. Within each group, conditions 1 and 2 allow us to use Theorem 1.1 to decompose K_h into factors of n-cycles. The unused edges of K_v form $K_{(h:u)}$. Conditions 3, 4, 5, and 6 allow us to use Theorem 1.2 to decompose $K_{(h:u)}$ into factors of m-cycles. ■

Chapter 3

When $m = 3$ and $n = 9$

3.1 Case I: $v \equiv 9 \pmod{18}$

On a graph with nine points, a 3- and 9-cycle URD is equivalent to a solution to the Hamilton-Waterloo problem with triangles and Hamilton-cycles. This was studied in [10], summarized below in the following theorem.

Theorem 3.1. *[10] Let $v = 3 + 6p$ and assume that $p \equiv 1 \pmod 3$, then there exists a $(3, v)$–$\mathsf{URD}(v; r, s)$ for every $0 \leq s \leq \frac{v-1}{2}$, $r = \frac{v-1}{2} - s$ except possibly $s = 1$.*
There is no $(3,9)$–$\mathsf{URD}(9; 3, 1)$.

The constructions used in this thesis are based on the existence of $(3,9)$–$\mathsf{URD}(9; r, s)$. The most basic construction revolves around partitioning K_v into parallel classes of K_9, then applying one of the four $(3,9)$–$\mathsf{URD}(9; r, s)$ decompositions to each copy of K_9. To obtain these parallel classes of K_9, we use 3–$\mathsf{RGDD}(h^u)$, when it exists. The following page contains the four $(3,9)$–$\mathsf{URD}(9; r, s)$ that exist.

Example 3.2. *A* $(3,9)$–URD$(9;r,s)$ *exists for* $(r,s) \in \{(4,0),(2,2),(1,3),(0,4)\}$.

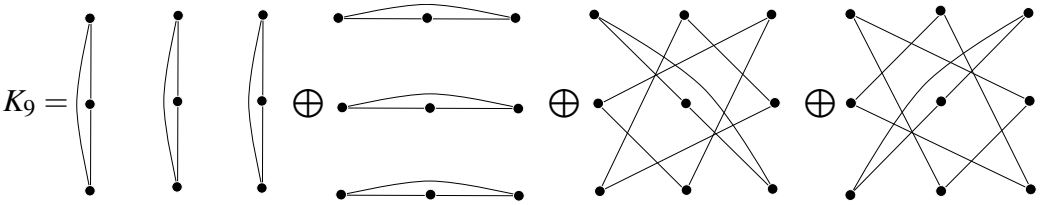

Figure 3.1: A $(3,9)$-URD$(9;4,0)$

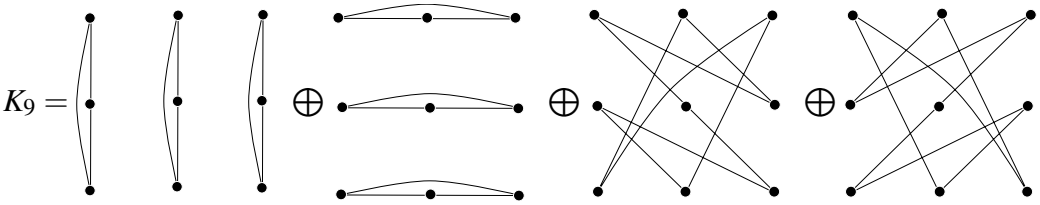

Figure 3.2: A $(3,9)$-URD$(9;2,2)$

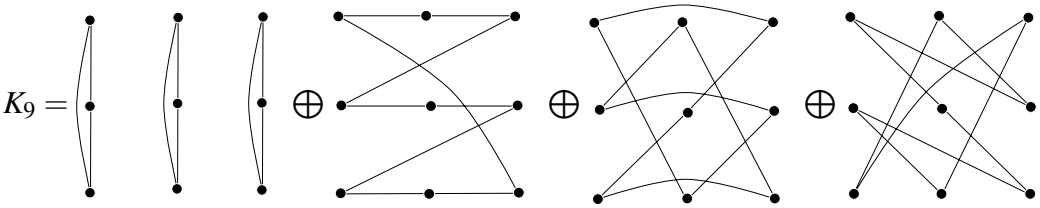

Figure 3.3: A $(3,9)$-URD$(9;1,3)$

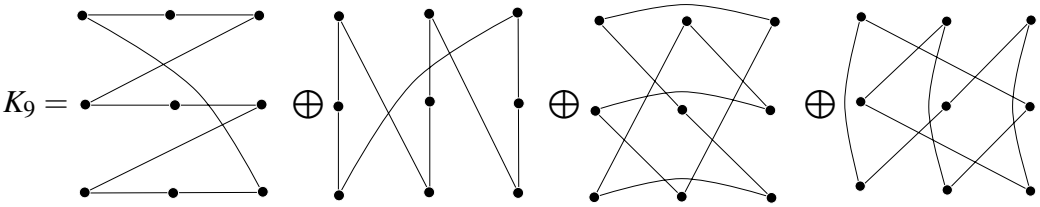

Figure 3.4: A $(3,9)$-URD$(9;0,4)$

Throughout these constructions, the complete equipartite graph $K_{(3:3)}$ will be as important as K_9. As such, the first thing we need to do is use Theorem 3.1 to solve a variant of the Hamilton-Waterloo problem on $K_{(3:3)}$

Lemma 3.3. *$K_{(3:3)}$ can be decomposed into r C_3-factors and s C_9-factors, where $(r,s) \in \{(0,3),(1,2),(3,0)\}$.*

Proof: By Theorem 3.1, K_9 can be decomposed into r' C_3-factors and s C_9-factors, where $(r',s) \in \{(1,3),(2,2),(4,0)\}$, as in Figures 3.1, 3.2, and 3.3. Each of these decompositions has at least one C_3-factor, say H. It is easy to see that $K_9 - H = K_{(3:3)}$, and that removing this factor will not affect the other three factors. So $r = r' - 1$, and $K_{(3:3)}$ can be decomposed into r C_3-factors and s C_9-factors, where $(r,s) \in \{(0,3),(1,2),(3,0)\}$. ■

We are now ready to detail the first of our major constructions. The complete, rigorous construction is to follow, but first: an outline.

The goal of the construction is to decompose K_v into parallel classes of $K_{(3:3)}$, then decompose those. "Parallel classes of $K_{(3:3)}$" is not mathematically rigorous or well defined, making this description suitable for an outline but not a proof.

We will begin by dividing v points into 3 large sets, then subdividing each of those large sets into $\frac{v}{9}$ subsets of 3 points each. Using these 3-point subsets as groups of the RGDD, we will construct identical 3–RGDD$(3^{(\frac{v}{9})})$s on each large set of $\frac{v}{3}$ points.

With the RGDD in place, we are free to start examining parallel-classes of K_9. We will reduce each copy of K_9 to $K_{(3:3)}$ to avoid re-using edges, then decompose each copy of $K_{(3:3)}$ as per Lemma 3.3.

After this process has been completed for each parallel class, we will have used nearly all the edges. The unused leftovers, at this stage, are the edges within and between corresponding groups of the RGDD. Since each group has size three, and there are three copies of each group, these leftover edges again form copies of K_9, which we can again decompose into some combination of triangle factors or C_9-factors.

Lemma 3.4. *There exists a* $(3,9)$–$\mathsf{URD}(v;r,s)$ *for any* $v \equiv 9 \pmod{18}$, $0 \leq r \leq \frac{v-1}{2}$ *and* $s = \frac{v-1}{2} - r$, *except possibly* $(r,s) = (\frac{v-3}{2}, 1)$.

Proof: Let $v \equiv 9 \pmod{18}$ and consider K_v. Begin by dividing the vertex set into three subsets of $\frac{v}{3}$ points each: G_1, G_2, and G_3. Using results from Theorem 1.5, there exists a 3–$\mathsf{RGDD}(3^{(\frac{v}{9})})$ on the $\frac{v}{3}$ points of G_1. An identical 3–$\mathsf{RGDD}(3^{(\frac{v}{9})})$ exists on each of G_2 and G_3. Divide the points of G_1, G_2 and G_3 according to the $\frac{v}{9}$ groups of the RGDD. The groups of G_i will be called $G_{i,1}, G_{i,2}, \ldots, G_{i,\frac{v}{9}}$ for $i = 1,2,3$. Each of these groups has 3 points, there are $\frac{v}{9}$ blocks in each parallel class, and there are $q = \frac{v-9}{6}$ parallel classes. Arbitrarily index the parallel classes $P_1, P_2, \ldots, P_q$.

Now consider any transverse pair of points $\{a,b\}$ with $a \in G_{1,\ell}$ and $b \in G_{1,k}$, $\ell \neq k$. This pair is contained in exactly one block of the RGDD on G_1, and this block is contained in exactly one parallel class P_p. Taking the union of this block and the identical blocks on the points of G_2 and G_3 gives us a set of 9 points. The subgraph of K_v induced by these 9 points is clearly K_9. Repeating this process for each other block of the parallel class gives us $(\frac{v}{9})$ copies of K_9. From each copy, remove the edges of the form $\{a,a'\}$ where $a \in G_{i,\ell}$ and $a' \in G_{j,\ell}$. The remaining edges form $K_{(3:3)}$ on each former copy of K_9. By Lemma 3.3, each copy of $K_{(3:3)}$ can be decomposed into r_p C_3-factors and s_p C_9-factors for $(r_p, s_p) \in \{(0,3),(1,2),(3,0)\}$. Choosing one such decomposition and applying it to each copy of $K_{(3:3)}$ will give us r_p factors of triangles and s_p factors of 9-cycles on the original graph, K_v.

Repeating this process on each of the parallel classes uses each edge not of the form $\{a,a'\}$ where $a \in G_{i,\ell}$ and $a' \in G_{j,\ell}$. Since we can choose different combinations of 3- and 9-cycles to decompose each parallel class, we have $\sum_{p=1}^{q} r_p = r_\alpha$ C_3-factors and $\sum_{p=1}^{q} s_p = s_\alpha$ C_9-factors at this stage. Necessarily, we have $r_\alpha + s_\alpha = \frac{v-9}{2}$.

The edges left are of the form $\{a,a'\}$ where $a \in G_{i,\ell}, a' \in G_{j,\ell}$, and $\ell \in \{1,2,\ldots,\frac{v}{9}\}$. Fixing $\ell = 1$ and examining all such edges, we see they form K_9. In fact, copies of K_9

are formed for each $\ell \in \{1,2,\ldots,\frac{v}{9}\}$, so we have $\frac{v}{9}$ copies of K_9. Each copy can each be decomposed into r_β C_3-factors and s_β C_9-factors for $(r_\beta, s_\beta) \in \{(4,0),(1,3),(2,2),(4,0)\}$ by Theorem 3.1, giving us the last four 2-factors.

■

At each stage of this construction, we can choose any number of 3- and 9-cycle factors except single 9-cycle factors. With these results, we can decompose K_{27} into any combination of $r = r_\alpha + r_\beta$ C_3-factors and $s = s_\alpha + s_\beta$ C_9-factors such that $r + s = \frac{v-1}{2}, s \neq 1$.

We now provide an example on $v = 27$ points to illustrate each step of the above construction.

Example 3.5. *The case when* $v = 27$.

First note that there will be a total of thirteen 2-factors. Label the vertices of K_{27} with elements of $\mathbb{Z}_{27}$ and divide the vertex set into three groups and nine subgroups. One such division might look like this: $G_{1,1} = \{0,3,6\}, G_{1,2} = \{1,4,7\}, G_{1,3} = \{2,5,8\}, G_{2,1} = \{9,12,15\}, \ldots, G_{3,3} = \{21,23,26\}$. Figure 3.5 shows the three parallel classes of the 3–RGDD(3^3) on the group G_1. Call these parallel classes P_1, P_2, and P_3. The parallel classes of G_2 can be obtained by adding nine to the label of each vertex in G_1 and the parallel classes of G_3 can be obtained by adding 18 to the label of each vertex in G_1.

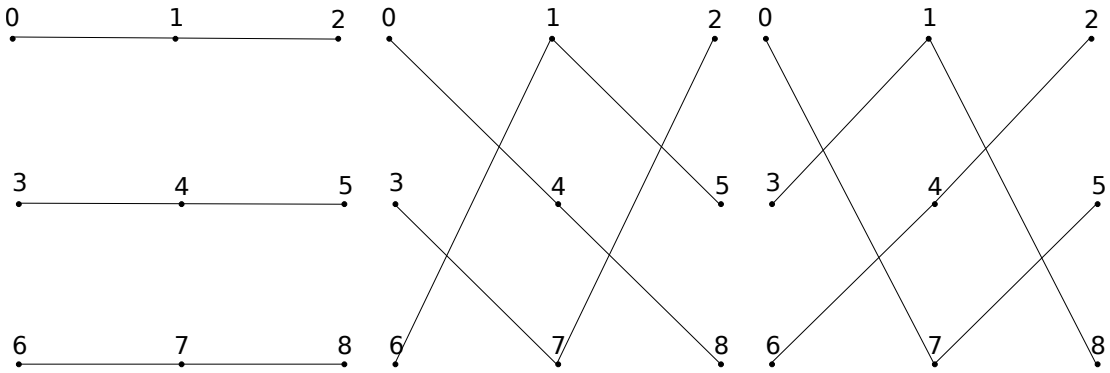

Figure 3.5: The three parallel classes of a 3-RGDD(3^3). From left to right, P_1, P_2, P_3.

Consider the arbitrarily chosen pair $\{2,7\}$ in G_1. This pair uniquely determines a block, $\{2,3,7\}$, in a uniquely determined parallel class, P_2 ($\{\{2,3,7\},\{1,5,6\},\{0,4,8\}\}$). The union of the block $\{2,3,7\}$ and its associated blocks in G_2 and G_3 ($\{11,12,16\}$ and $\{20,21,25\}$, respectively) gives us nine points. The subgraph of K_{27} induced on these nine points is K_9. We remove from this graph all edges between points of associated subgroups. In this case, we remove the following edges: $\{2,11\}$, $\{11,20\}$, $\{2,20\}$, $\{3,12\}$, $\{12,21\}$, $\{3,21\}$, $\{7,16\}$, $\{16,25\}$, $\{7,25\}$. After removing these edges, we are left with $K_{(3:3)}$, as shown in Figure 3.6.

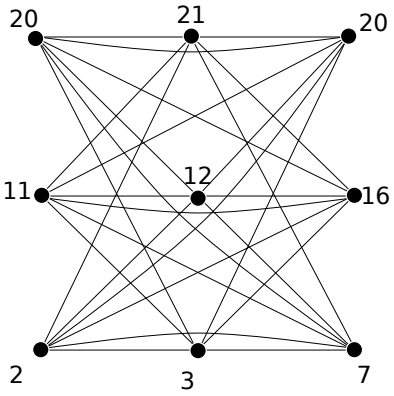

Figure 3.6: The copy of $K_{(3:3)}$ given by the block $\{2,3,7\}$

Repeating this process with the blocks $\{1,5,6\}$ and $\{0,4,8\}$ will give us two additional, identical, and disjoint copies of $K_{(3:3)}$. Using Lemma 3.3, each of these three copies of $K_{(3:3)}$ can be decomposed into three factors of 3-cycles and zero factors of 9-cycles ($r_2 = 3, s_2 = 0$), one factor of 3-cycles and two factors of 9-cycles ($r_2 = 1, s_2 = 2$), or zero factors of 3-cycles and three factors of 9-cycles ($r_2 = 0, s_2 = 3$). We choose one such combination of C_3-factors and C_9-factors and apply it to each of our three copies of $K_{(3:3)}$. This gives us three 2-factors.

From here we repeat this process, arbitrarily choosing new pairs of vertices on G_1 that have not yet been accounted for in parallel class P_2. The pair $\{1,2\}$ is in the unique block $\{1,2,3\}$ in the parallel class P_1 ($\{\{1,2,3\},\{4,5,6\},\{7,8,9\}\}$). Each of these blocks, when

joined with its associated blocks in G_2 and G_3, again gives us nine points, on which we form $K_{(3:3)}$ graphs and decompose them into C_3-factors and C_9-factors according to Lemma 3.3. A third run through this process using the parallel class P_3 $(\{\{0,5,7\},\{1,3,8\},\{2,4,6\}\})$ will use the last of the transverse pairs in G_1. This gives us six more 2-factors in any combination of (r_α, s_α) where $(r_\alpha, s_\alpha) = \sum_{p=1}^{3}(r_p, s_p)$. Necessarily, $r_\alpha + s_\alpha = 9$, and s_α cannot be 1.

Now we form three new sets of nine points each using associated groups of the RGDDs instead of associated blocks. The first such set of nine is $\{0,3,6\}\cup\{9,12,15\}\cup\{18,21,24\}$. The subgraph of K_{27} induced by this set is K_9. The other sets that also induce K_9 subgraphs are $\{1,4,7\}\cup\{10,13,16\}\cup\{19,22,25\}$ and $\{2,5,8\}\cup\{11,14,17\}\cup\{20,23,26\}$. K_9 can be decomposed, as in Example 3.2, into four factors of 3-cycles and zero factors of 9-cycles ($r_\beta = 4, s_\beta = 0$, Figure 3.1), two factors of 3-cycles and two factors of 9-cycles ($r_\beta = 2, s_\beta = 2$, Figure 3.2), one factor of 3-cycles and three factors of 9-cycles ($r_\beta = 1, s_\beta = 3$, Figure 3.3), or zero factors of 3-cycles and four factors of 9-cycles ($r_\beta = 0, s_\beta = 4$, Figure 3.4). Choosing one such combination of factors and applying it to each copy of K_9 gives us the last four 2-factors, for a total of thirteen 2-factors, in any combination except a single factor of 9-cycles.

The case when $s = 1$ cannot be handled by Lemma 3.4 because there is no $(3,9)$–$\mathsf{URD}(9;3,1)$. This case is considerably more difficult. This thesis handles only a fraction of these cases using a recursive construction and a base case. The base case, a $(3,9)$–$\mathsf{URD}(27;12,1)$, was found by computer search using difference methods in Mathematica 8. The code for this program can be found in Chapter 4. The recursive construction allows for v to be any odd multiple of 27.

Lemma 3.6. *There exists a* $(3,9)$–$\mathsf{URD}(27;12,1)$.

Proof: Consider K_{27}. Partition the points into three groups of size nine each. Label the groups G_i, $i \in \{1,2,3\}$. Label the vertices in each group by the elements of $\mathbb{Z}_9 \times \{1,2,3\}$ such that the subscript from $\{1,2,3\}$ matches the group the vertex is in.

Traditionally, we think of an edge as being defined by two vertices. However, with each vertex numbered, each edge can also be labeled with the difference (modulo nine) between the two vertices it connects. This way, we can think of an edge as being defined by a vertex label and an edge label.

Applied to our representation of K_{27} as elements of $\mathbb{Z}_9 \times \{1,2,3\}$, this idea of edges as differences can be quite powerful. Our goal is to use each edge exactly once. We will do this by constructing several base blocks that use distinct differences. When we develop this block modulo nine, It will not re-use any edges because each difference is distinct.

We have to be careful when we talk about differences being distinct. The edge $2_1, 4_1$ seems to have a difference of 2, and the edge $4_1, 2_1$ seems to have difference -2, but we know these are the same edge. Within each group, there are only four differences to cover ($\pm 1, \pm 2, \pm 3$, and ± 4). Considering edges between different groups, we find that the idea of difference is not well defined.

Because there are only three groups, transverse edge will have the form α_i, β_{i+1}, where the index $i+1$ is taken modulo 3. Define the difference associated with edge α_i, β_{i+1} to be $(\beta_{i+1} - \alpha_i) \pmod 9$. Here, there are nine differences to cover: (0, 1, 2, 3, 4, 5, 6, 7, and 8). We call these the mixed differences.

The first set of blocks we consider, $\mathcal{X}$, is a full factor of 3-cycles:

$$\{0_1, 1_1, 0_2\}, \{2_1, 4_1, 5_2\}, \{3_1, 6_1, 1_2\},$$
$$\{2_2, 3_2, 0_3\}, \{4_2, 7_2, 3_3\}, \{6_2, 8_2, 1_3\},$$
$$\{2_3, 5_3, 7_1\}, \{4_3, 8_3, 8_1\}, \{6_3, 7_3, 5_1\}.$$

$\mathcal{X}$ is given by Figure 3.7.

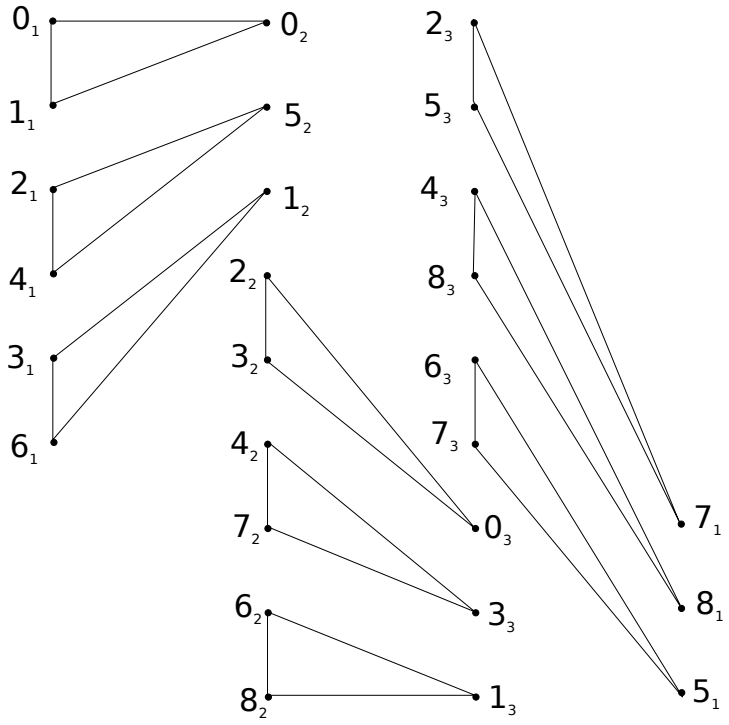

Figure 3.7: $\mathcal{X}$, a factor of 3-cycles that can be developed modulo nine.

Because $\mathcal{X}$ is itself a C_3-factor and no difference is repeated, developing it modulo nine gives us nine edge-disjoint C_3-factors. Now we examine what differences have been used. Within G_1, we have used edges representing differences of $\pm 1, \pm 2$, and ± 3. This leaves only differences of ± 4 in G_1. Within G_2, we have used edges representing differences of $\pm 1, \pm 3$, and ± 2. This leaves only differences of ± 4 in G_2. Within G_3, we have used edges representing differences of $\pm 3, \pm 4$, and ± 1. This leaves only differences of ± 2 in G_3.

We use these differences to create our second set of blocks, $\mathcal{Y}$, which consists of the following 9-cycles:

$$\{0_1,4_1,8_1,3_1,7_1,2_1,6_1,1_1,5_1\},$$
$$\{0_2,4_2,8_2,3_2,7_2,2_2,6_2,1_2,5_2\},$$
$$\{0_3,2_3,4_3,6_3,8_3,1_3,3_3,5_3,7_3\}.$$

$\mathcal{Y}$ is a full factor of 9-cycles. It does not need to be developed.

Lastly, we consider the unused edges between groups. Between G_1 and G_2, mixed differences 0, 8, 3, 1, 7, and 4 have been used, leaving 2, 5 and 6. Between G_2 and G_3, mixed differences 7, 6, 8, 5, 4, and 2 have been used, leaving 0, 1 and 3. Between G_3 and G_1, mixed differences 5, 2, 4, 0, 8, and 7 have been used, leaving 1, 3, and 6. We will make three factors of 3-cycles out of these nine unused differences.

First we use the difference 6 between G_1 and G_2, the difference 0 between G_2 and G_3, and the difference 3 between G_3 to G_1. The base block is $\mathcal{Z}_1 = \{0_1,6_2,6_3\}$. Next, we use the difference 2 between G_1 to G_2, the difference 1 between G_2 to G_3, and the difference 6 between G_3 to G_1. The base block is $\mathcal{Z}_2 = \{0_1,2_2,3_3\}$. Lastly, we use the difference 5 between G_1 and G_2, the difference 3 between G_2 and G_3, and the difference 1 between G_3 and G_1. The base block is $\mathcal{Z}_3 = \{0_1,5_2,8_3\}$. Each of these three base blocks, when developed modulo nine, will generate a full C_3-factor.

We have nine C_3-factors from developing $\mathcal{X}$, a single C_9-factor from $\mathcal{Y}$, and one more C_3-factor from each of $\mathcal{Z}_1, \mathcal{Z}_2$, and $\mathcal{Z}_3$. This uses each edge exactly once and gives us twelve C_3-factors and one C_9-factor.

■

The following Lemma combines the existence of a $(3,9)$–$\mathsf{URD}(27;12,1)$ with Liu's result regarding the Oberwolfach Problem on equipartite graphs, Theorem 1.2.

Lemma 3.7. *There exists a* $(3,9)$–$\mathsf{URD}(v;\frac{v-3}{2},1)$ *for any* $v \equiv 27 \pmod{54}$.

Proof: Since $v \equiv 27 \pmod{54}$, we have that $27|v$ and that $q = \frac{v}{27}$ is odd. Partition the points of K_v into q groups of 27 points each. Within each group of 27 points, we use Lemma 3.6 to decompose K_{27} into triangle factors and a single C_9-factor. We are left with all edges between groups, which form $K_{(27:q)}$. Using Theorem 1.2, this graph can be decomposed into 3-cycle factors. This gives us a full 2-factorization of K_v into 3-cycles and 9-cycles with exactly one 9-cycle factor. ■

3.2 Case II: $v \equiv 0 \pmod{18}$

The construction of URDs with an even number of vertices is similar in almost every way. Again with intentions of resolving V, the vertex set of K_v, into parallel classes of $K_{(3:3)}$ graphs, we divide the vertex set into three sets and divide each of those sets into smaller subsets. In the even case, it is most convenient to use subsets of size 6. Theorem 1.5 again guarantees the existence of an RGDD, this time a 3–$\mathsf{RGDD}(6^{(\frac{v}{18})})$, as long as $\frac{v}{18} = u \geq 4$, on each of the three sets. We use the parallel classes of these RGDDs to get parallel classes of K_9, which we again reduce to $K_{(3:3)}$ and decompose as per Lemma 3.3.

The biggest difference between the even and odd cases comes in the form of the leftover vertices after each parallel class of $K_{(3:3)}$ has been decomposed. The remaining edges again are those within and between corresponding groups of the RGDDs, though in this case, the size of each group was six, which leaves copies of K_{18} as the leftovers. At first glance, this seems like a problem because a $(3,9)$–$\mathsf{URD}(18;r',s')$ is not known to exist for every pair of (r',s'). To finish the construction, we must look back to the Oberwolfach Problem. Theorem 1.1 allows us to decompose K_{18} into eight C_3-factors or eight C_9-factors. This rigidity seems detrimental, but is easily overcome by the flexibility provided to us in the decomposition of the parallel classes of $K_{(3:3)}$. Lemma 3.8, below, details the flexibility of

the choices provided to us by these parallel classes.

The notation introduced in Lemma 3.4 for keeping track of (r,s) using subscripts will be crucial to understanding the following Lemma. For the following result, we have $(r,s) = (r_\beta, s_\beta) + (r_\alpha, s_a)$, where $(r_\beta, s_\beta) \in Y = \{(0,8),(8,0)\}$ and $(r_\alpha, s_\alpha) = \sum_{p=1}^{\frac{v-18}{6}} (r_p, s_p)$, $(r_p, s_p) \in X = \{(0,3),(1,2),(3,0)\}\ \forall\ 0 \leq p \leq \frac{v-18}{6}$. Each (r_p, s_p) comes from the choice of decompositions in the parallel class p, and they sum to (r_α, s_α). The pair (r_β, s_β) comes from our choice of decomposing the leftover copies of K_{18} into eight C_3-factors or eight C_9-factors.

Lemma 3.8. *Let $v \equiv 0 \pmod{18}, v \geq 72$. Any pair (r,s) of integers with the properties $r,s \geq 0$, $r+s = \frac{v-2}{2}$, $s \neq 1$ can be obtained by the sum of $q = \frac{v-18}{6}$ elements from the set $X = \{(0,3),(1,2),(3,0)\}$ and one element from the set $Y = \{(0,8),(8,0)\}$.*

Proof: Let $v \equiv 0 \pmod{18}, v \geq 72$, set $q = \frac{v-18}{6}$, and let $r,s \in \mathbb{Z}$ be given such that $r,s \geq 0$, $r+s = \frac{v-2}{2}$, $s \neq 1$. Further let $X = \{(0,3),(1,2),(3,0)\}$ and $Y = \{(0,8),(8,0)\}$.

Note that each ordered pair in X sums to 3, and each ordered pair in Y sums to 8. A quick calculation will reveal that $(3)(\frac{v-18}{6}) + 8 = \frac{v-2}{2}$. So no matter which pairs we choose, the property that $r+s = \frac{v-2}{2}$ will always be satisfied. Therefore, it will suffice to show that any integer $0 \leq s \leq \frac{v-2}{2}, s \neq 1$ can be obtained by the sum of $q = \frac{v-18}{6}$ elements from the set $X' = \{3,2,0\}$ and one element from the set $Y' = \{8,0\}$. The following step-by-step proof is constructive.

1. If $s \geq 10$, choose 8 from Y' and set $(r_\beta, s_\beta) = (0,8)$. If not, choose 0 from Y' and set $(r_\beta, s_\beta) = (8,0)$. Now we must choose q numbers from X' such that they sum to $s - s_\beta$.

2. If $s - s_\beta$ is odd, choose 3 from X', set $(r_q, s_q) = (0,3)$, and set $s' = s - s_\beta - 3$. Otherwise, make no choice from X', do not assign values to (r_q, s_q), and set $s' = s - s_\beta$.

3. Let $w = \lfloor \frac{s'}{6} \rfloor$. Choose 3 from X' $2w$ times. For each index p between 1 and $2w$ inclusive, set $(r_p, s_p) = (0,3)$.

4. If $s' \equiv 4 \pmod 6$, choose 2 from X' twice.
   Set $(r_{2w+1}, s_{2w+1}) = (r_{2w+2}, s_{2w+2}) = (1,2)$

5. If $s' \equiv 2 \pmod 6$, choose 2 from X' once. Set $(r_{2w+1}, s_{2w+1}) = (1,2)$

6. Each other choice from X' is zero, and each yet unassigned $(r_p, s_p) = (3,0)$.

It is easy to see that this algorithm will choose integers from X' and Y' that sum to s. It is left to show that it uses the correct number of choices from X'. Because Step 6 fills all yet unused choices with zeros, we cannot make too few choices from X. The only risk is taking too many choices from X. Assume towards contradiction that this algorithm chooses $\frac{v-18}{6} + 1$ numbers from X. First note that if $s < 10$, then the algorithm makes at most 4 non-zero choices, and since $v \geq 72$, we already have a contradiction. So assume $s \geq 10$, and our choice from Y must have been 8. Since zeros are only chosen to fill unused choices, each choice made from X must be a two or a three. Assume two choices from X were twos. The remaining $\frac{v-18}{6} - 1$ choices must have been threes. Adding up all these choices, we get $1(8) + 2(2) + 3(\frac{v-18}{6} - 1) = \frac{v}{2}$, which contradicts our assumption that $s \leq \frac{v-2}{2}$. Assume one choice from X was a two. The remaining $\frac{v-18}{6}$ choices must have been threes. Adding up all these choices, we get $1(8) + 1(2) + 3(\frac{v-18}{6}) = \frac{v+2}{2}$, which again contradicts our assumption that $s \leq \frac{v-2}{2}$. Finally, assume that zero choices from X were twos. The remaining $\frac{v-18}{6} + 1$ choices must have been threes. Adding up all these choices, we get $1(8) + 0(2) + 3(\frac{v-18}{6} + 1) = \frac{v+4}{2}$, which still contradicts our assumption that $s \leq \frac{v-2}{2}$. Therefore, this algorithm cannot choose more or less than $\frac{v-18}{6}$ elements of X.

■

We now have all the tools in place to begin a rigorous look at the construction of a $(3,9)$–$\mathsf{URD}(v;r,s)$ where v is even.

Lemma 3.9. *There exists a* $(3,9)$–$\mathsf{URD}(v;r,s)$ *for any* $v \equiv 0 \pmod{18}$, $v \geq 72$, $0 \leq r \leq \frac{v-2}{2}$ *and* $s = \frac{v-2}{2} - r$, *except possibly* $(r,s) = (\frac{v-4}{2}, 1)$.

Proof:

Let $v \equiv 0 \pmod{18}, v \geq 72$, and let (r,s) be given such that $r+s = \frac{v-2}{2}$ and $s \neq 1$. First, we apply the algorithm described in the proof of Lemma 3.8 to determine what choices we should make for (r_p, s_p) for each $p \in \{1,2,\ldots,\frac{v-18}{6}\}$.

Because $v \geq 72$, we have that $u = \frac{v}{18} \geq 4$. Begin by dividing the vertex set of K_v into three subsets of $\frac{v}{3}$ points each: G_1, G_2, and G_3. Using results from Theorem 1.5, there exists a 3–$\mathsf{RGDD}(6^u)$ on the $\frac{v}{3}$ points of G_1. An identical 3–$\mathsf{RGDD}(6^u)$ exists on each of G_2 and G_3. Divide the points of G_1, G_2 and G_3 according to the u groups of the RGDD. The groups of G_i will be called $G_{i,1}, G_{i,2}, \ldots G_{i,u}$ for $i = 1,2,3$. Each of these groups has 6 points with $\frac{v}{9}$ blocks in each parallel class and $q = \frac{v-18}{6}$ parallel classes. Arbitrarily index the parallel classes by $P_1, P_2, \ldots, P_q$.

Now consider any transverse pair of points $\{a,b\}$ with $a \in G_{1,\ell}$ and $b \in G_{1,k}$, $\ell \neq k$. This pair is contained in exactly one block of the RGDD on G_1, and this block is contained in exactly one parallel class, p. Taking the union of this block and the identical blocks on the points of G_2 and G_3 gives us a set of 9 points. The subgraph of K_v induced by these 9 points is clearly K_9. Repeating this process for each other block of the parallel class gives us $\frac{v}{9}$ copies of K_9. From each copy, remove the edges of the form $\{a,a'\}$ where $a \in G_{i,\ell}$ and $a' \in G_{j,\ell}$. The remaining edges of the K_9 subgraph form $K_{(3:3)}$. By Lemma 3.3, each copy of $K_{(3:3)}$ can be decomposed into r_p C_3-factors and s_p C_9-factors for $(r_p, s_p) \in \{(0,3),(1,2),(3,0)\}$. Choosing one such decomposition and applying it to each copy of $K_{(3:3)}$ will give us r_p factors of triangles and s_p factors of 9-cycles on the original graph, K_v.

Repeating this process on each of the parallel classes uses each edge not of the form $\{a,a'\}$ where $a \in G_{i,\ell}$ and $a' \in G_{j,\ell}$. Since we can choose different combinations of 3- and 9-cycles to decompose each parallel class, we have $\sum_{p=1}^{q} r_p = r_\alpha$ C_3-factors and $\sum_{p=1}^{q} s_p = s_\alpha$ C_9-factors , where necessarily $r_\alpha + s_\alpha = \frac{v-18}{2}$.

The only remaining edges are of the form $\{a,a'\}$ where $a \in G_{i,\ell}, a' \in G_{j,\ell}$, and $\ell \in \{1,2,3\}$. Fixing $\ell = 1$ and examining all such edges, we see they form K_{18}. In fact, copies of K_{18} are formed for each $\ell = 1,2,\ldots,u$, so we have u copies of K_{18}. On each copy, we use results from the Oberwolfach Problem given by Theorem 1.1 to decompose the edge set of K_{18} into eight C_3-factors and a 1-factor or eight C_9-factors and a 1-factor. This gives us $(r_\beta, s_\beta) \in \{(8,0),(0,8)\}$. Now, each edge in K_v has been accounted for by C_3-factors, C_9-factors, or a 1-factor, and by Lemma 3.8, $(r_\alpha, s_\alpha) + (r_\beta, s_\beta) = (r,s)$.

■

Lemma 3.9 handles all cases of $v \equiv 0 \pmod{18}$ so long as $v \geq 72$. If we attempt to apply the same construction to $v = 18$, 36, or 54, we will find that the required RGDD does not exist.

Chapter 4

Conclusions

Between Lemma 3.4, Lemma 3.7, and Lemma 3.9, we have shown that the necessary conditions for the existence of a $(3,9)$–$\mathsf{URD}(v;r,s)$ are sufficient with few exceptions. The remaining open cases are

1. $v = 18$,
2. $v = 36$,
3. $v = 54$, and
4. $s = 1$, when v is not equivalent to $27 \pmod{54}$.

Theorem 4.1 summarizes these results.

Theorem 4.1. *Let* $v \equiv 0 \pmod 9$, $v \neq 18, 36, 54$. *Then there exists a* $(3,9)$–$\mathsf{URD}(v;r,s)$ *for all pairs of positive integers* (r,s) *such that* $r+s = \frac{v-1}{2}$ *if* v *is odd or* $r+s = \frac{v-2}{2}$ *is* v *is even, possibly except* $s = 1$ *when* $v \not\equiv 27 \pmod{54}$.

Proof: Let $v \equiv 0 \pmod 9$, $v \neq 18, 36, 54$ be given. If v is odd, we apply Lemma 3.4 to construct a solution. If v is even, we apply Lemma 3.9 to construct a solution. If $v \equiv 27 \pmod{54}$ and $s = 1$, we apply Lemma 3.7 to construct a solution. ■

Bibliography

[1] P. Adams, E. Billington, D. Bryant, and S. El-Zanati, *On the Hamilton-Waterloo problem*, Graphs and Combinatorics **18** (2002), 31–51.

[2] B. Alspach and R. Haggkvist, *Some observations on the Oberwolfach problem*, Journal of Graph Theory **9** (1985), 177–187.

[3] B. Alspach, P. Schellenberg, D.R. Stinson, and D. Wagner, *The Oberwolfach problem and factors of uniform length*, Journal of Combinatorial Theory, Ser. A **52** (1989), 20–43.

[4] C. Colbourn and J. Dinitz, *The CRC handbook of combinatorial designs*, Discrete Mathematics, CRC Press, Boca Raton, 2006.

[5] P. Danziger, G. Quattrocchi, and B. Stevens, *The Hamilton-Waterloo problem for cycle sizes 3 and 4*, J. Combin. Des. **17** (2009), no. 4, 342–352.

[6] R. Diestel, *Graph theory*, Graduate Texts in Mathematics, vol. 173, Springer, New York, 2000.

[7] J.H. Dinitz and A.C.H. Ling, *The Hamilton-Waterloo problem: the case of triangle-factors and one Hamilton cycle*, J. Combin. Des. **17** (2009), no. 2, 160–176.

[8] H. Fu and K. Huang, *The Hamilton-Waterloo problem for two even cycles factors*, Taiwanese J. Math. **12** (2008), no. 4, 933–940.

[9] D.G. Hoffman and P.J. Schellenberg, *The existence of C_k-factorizations of $K_{2n} - F$*, Discrete Math **97** (1991), 243–250.

[10] P. Horak, R. Nedela, and A. Rosa, *The Hamilton-Waterloo problem: the case of Hamilton cycles and triangle-factors*, Discrete Math. **284** (2004), no. 1-3, 181–188.

[11] M. Keranen and S. Özkan, *The Hamilton-Waterloo problem with 4-cycles and a single factor of n-cycles*, Graphs and Combinatorics, (submitted).

[12] A.C.H. Ling and J.H. Dinitz, *The Hamilton-Waterloo problem with triangle-factors and Hamilton cycles: the case $n \equiv 3 \pmod{18}$*, J. Combin. Math. Combin. Comput. **70** (2009), 143–147.

[13] J. Liu, *The equipartite Oberwolfach problem with uniform tables*, J. Combin. Theory Ser. A **101** (2003), no. 1, 20–34.

[14] R.S. Rees, *Two new direct product-type constructions for resolvable group-divisible designs*, Journal of Combinatorial Designs **1** (1993), 15–26.

[15] D. Stinson, *Combinatorial designs*, Springer-Verlag, New York, 2004.

Appendix

The Program

This program was written in Mathematica 8 to find solutions to the $(3,9)$–URD$(27;12,1)$ case. The solution given in Lemma 3.6 is the first this program returned. It is sparsely commented with explainations.

This block of code creates the pairs and triples of points that we'll be working with. The output checks the dimensions of each.

Input:

nineCtwo = Permutations[{0,1,2,3,4,5,6,7,8},{2}];

nineCtwo = Union[Table[Sort[nineCtwo[[*i*]]],{*i*,1,Length[nineCtwo]}]];

nineCtwo//MatrixForm;

Dimensions[nineCtwo]

nineCthree = Permutations[{0,1,2,3,4,5,6,7,8},{3}];

Dimensions[nineCthree]

Output:

$\{36,2\}$

$\{504,3\}$

doubles is a list of triples of indices of **nineCtwo**. Each triple of pairs has no overlap (that is, each triple represents six distinct points and the three differences used must include

the difference ± 3). Again, the output checks the dimensions of this structure.

Input:

```
doubles = Flatten[Drop[Reap[
For[i = 1, i ≤ Length[nineCtwo], i++,
For[j = i + 1, j ≤ Length[nineCtwo], j++,
If[Length[Union[nineCtwo[[i]], nineCtwo[[j]]]] == 4,
For[k = j + 1, k ≤ Length[nineCtwo], k++,
If[Length[Union[nineCtwo[[i]], nineCtwo[[j]], nineCtwo[[k]]]] == 6,
If[(MemberQ[{3, 6}, Mod[nineCtwo[[i, 1]] − nineCtwo[[i, 2]], 9]] ∨
MemberQ[{3, 6}, Mod[nineCtwo[[j, 1]] − nineCtwo[[j, 2]], 9]] ∨
MemberQ[{3, 6}, Mod[nineCtwo[[k, 1]] − nineCtwo[[k, 2]], 9]]) ∧
(Length[Union[{Mod[nineCtwo[[i, 1]] − nineCtwo[[i, 2]], 9],
Mod[nineCtwo[[j, 1]] − nineCtwo[[j, 2]], 9],
Mod[nineCtwo[[k, 1]] − nineCtwo[[k, 2]], 9]}]] == 3),
Sow[{i, j, k}], Null], Null]], Null]]], 1], 2];
doubles//MatrixForm;
Dimensions[doubles]
```

Output:

$\{449, 3\}$

tri1 attempts to match each pair in each triple in **doubles** (representing an interior edge of G_1) to a point in G_2, forming three triangles. The six cross differences must be distinct. Again, the only output is the size of **tri1**.

Input:

```
tri1 = Flatten[Drop[Reap[
For[i = 1, i ≤ Length[doubles], i++,
```

```
For[j = 1, j ≤ Length[nineCthree], j++,
If[Length[Union[{Mod[nineCthree[[j, 1]] − nineCtwo[[doubles[[i, 1]], 1]], 9],
Mod[nineCthree[[j, 1]] − nineCtwo[[doubles[[i, 1]], 2]], 9],
Mod[nineCthree[[j, 2]] − nineCtwo[[doubles[[i, 2]], 1]], 9],
Mod[nineCthree[[j, 2]] − nineCtwo[[doubles[[i, 2]], 2]], 9],
Mod[nineCthree[[j, 3]] − nineCtwo[[doubles[[i, 3]], 1]], 9],
Mod[nineCthree[[j, 3]] − nineCtwo[[doubles[[i, 3]], 2]], 9]}]] == 6,
Sow[{i, j}], Null]]]], 1], 2];
Dimensions[tri1]
tri1[[1;;50]];
```

Output:

$\{26046, 2\}$

We see now that at this stage, there are 26046 possible solutions. The following code attempts to complete each one. First it finds a triple of edges in G_2 with distinct differences, one of which is 3, none of whose points overlap with the points of **tri1**. It stores these solutions as a list of indices of **nineCtwo**, called **dbl2**. Then it finds three more points from G_3 that, when connected with the edges from **dbl2**, yield six distinct cross-differences. These working sets of 3 points in G_3 are stored, again as a list of indices, in **tri3**. The program then attempts to form three edges in G_3, again with three distinct differences that include 3, whose cross-differences with the three unused points of G_1 form the original set of three edges given in **doubles** are all distinct. Lastly, **a**, **b**, and **c** are used to check whether all the unused cross-differences can be made to sum to multiples of nine.

Input:

```
dblorig = {};
```

```
nine = {0, 1, 2, 3, 4, 5, 6, 7, 8};
Monitor[
For[q = 1, q ≤ Length[tri1], q++,
dbl2 = Flatten[Drop[Reap[
For[i = 1, i ≤ Length[doubles], i++,
If[Length[Union[nineCtwo[[doubles[[i, 1]]]], nineCtwo[[doubles[[i, 2]]]],
nineCtwo[[doubles[[i, 3]]]], nineCthree[[tri1[[q, 2]]]]] == 9,
Sow[i], Null]]], 1]];
tri2 = Flatten[Drop[Reap[
For[i = 1, i ≤ Length[dbl2], i++,
For[j = 1, j ≤ Length[nineCthree], j++,
If[Length[Union[{
Mod[nineCthree[[j, 1]] − nineCtwo[[doubles[[dbl2[[i]], 1]], 1]], 9],
Mod[nineCthree[[j, 1]] − nineCtwo[[doubles[[dbl2[[i]], 1]], 2]], 9],
Mod[nineCthree[[j, 2]] − nineCtwo[[doubles[[dbl2[[i]], 2]], 1]], 9],
Mod[nineCthree[[j, 2]] − nineCtwo[[doubles[[dbl2[[i]], 2]], 2]], 9],
Mod[nineCthree[[j, 3]] − nineCtwo[[doubles[[dbl2[[i]], 3]], 1]], 9],
Mod[nineCthree[[j, 3]] − nineCtwo[[doubles[[dbl2[[i]], 3]], 2]], 9]
}]] == 6,
Sow[{i, j}], Null]]], 1], 2];

Print[Timing[Monitor[
For[p = 1, p ≤ Length[tri2], p++,
dbl3 = Flatten[Drop[Reap[
For[i = 1, i ≤ Length[doubles], i++,
If[Length[Union[nineCtwo[[doubles[[i, 1]]]], nineCtwo[[
```

```
doubles[[i,2]]]],nineCtwo[[doubles[[i,3]]]],nineCthree[[
tri2[[p,2]]]]]] == 9,
Sow[i],Null]]],1]];
tri3 = Flatten[Drop[Reap[
For[i = 1,i ≤ Length[dbl3],i++,
For[j = 1,j ≤ Length[nineCthree],j++,
If[Length[Union[{
Mod[nineCthree[[j,1]] − nineCtwo[[doubles[[dbl3[[i]],1]],1]],9],
Mod[nineCthree[[j,1]] − nineCtwo[[doubles[[dbl3[[i]],1]],2]],9],
Mod[nineCthree[[j,2]] − nineCtwo[[doubles[[dbl3[[i]],2]],1]],9],
Mod[nineCthree[[j,2]] − nineCtwo[[doubles[[dbl3[[i]],2]],2]],9],
Mod[nineCthree[[j,3]] − nineCtwo[[doubles[[dbl3[[i]],3]],1]],9],
Mod[nineCthree[[j,3]] − nineCtwo[[doubles[[dbl3[[i]],3]],2]],9]}]] == 6,
Sow[{i,j}],Null]]]],1],2];
For[i = 1,i ≤ Length[tri3],i++,
a = Complement[nine,
Union[{Mod[nineCthree[[tri1[[q,2]],1]] − nineCtwo[[doubles[[tri1[[q,1]],1]],1]],9],
Mod[nineCthree[[tri1[[q,2]],1]] − nineCtwo[[doubles[[tri1[[q,1]],1]],2]],9],
Mod[nineCthree[[tri1[[q,2]],2]] − nineCtwo[[doubles[[tri1[[q,1]],2]],1]],9],
Mod[nineCthree[[tri1[[q,2]],2]] − nineCtwo[[doubles[[tri1[[q,1]],2]],2]],9],
Mod[nineCthree[[tri1[[q,2]],3]] − nineCtwo[[doubles[[tri1[[q,1]],3]],1]],9],
Mod[nineCthree[[tri1[[q,2]],3]] − nineCtwo[[doubles[[tri1[[q,1]],3]],2]],9]}]];
a1 = Sum[a[[j]],{j,1,3}];

b = Complement[nine,
Union[{Mod[nineCthree[[tri2[[p,2]],1]]−
```

```
nineCtwo[[doubles[[dbl2[[tri2[[p,1]]]],1]],1]],9],
Mod[nineCthree[[tri2[[p,2]],1]]-nineCtwo[[doubles[[dbl2[[tri2[[p,1]]]],1]],2]],9],
Mod[nineCthree[[tri2[[p,2]],2]]-nineCtwo[[doubles[[dbl2[[tri2[[p,1]]]],2]],1]],9],
Mod[nineCthree[[tri2[[p,2]],2]]-nineCtwo[[doubles[[dbl2[[tri2[[p,1]]]],2]],2]],9],
Mod[nineCthree[[tri2[[p,2]],3]]-nineCtwo[[doubles[[dbl2[[tri2[[p,1]]]],3]],1]],9],
Mod[nineCthree[[tri2[[p,2]],3]]-nineCtwo[[doubles[[dbl2[[tri2[[p,1]]]],3]],2]],9]}]];
b1=Sum[b[[j]],{j,1,3}];

c=Complement[nine,
Union[{Mod[nineCthree[[tri3[[i,2]],1]]-
nineCtwo[[doubles[[dbl3[[tri3[[i,1]]]],1]],1]],9],
Mod[nineCthree[[tri3[[i,2]],1]]-nineCtwo[[doubles[[dbl3[[tri3[[i,1]]]],1]],2]],9],
Mod[nineCthree[[tri3[[i,2]],2]]-nineCtwo[[doubles[[dbl3[[tri3[[i,1]]]],2]],1]],9],
Mod[nineCthree[[tri3[[i,2]],2]]-nineCtwo[[doubles[[dbl3[[tri3[[i,1]]]],2]],2]],9],
Mod[nineCthree[[tri3[[i,2]],3]]-nineCtwo[[doubles[[dbl3[[tri3[[i,1]]]],3]],1]],9],
Mod[nineCthree[[tri3[[i,2]],3]]-nineCtwo[[doubles[[dbl3[[tri3[[i,1]]]],3]],2]],9]}]];
c1=Sum[c[[j]],{j,1,3}];

If[Length[Union[nineCtwo[[doubles[[tri1[[q,1]],1]]]],
nineCtwo[[doubles[[tri1[[q,1]],2]]]],nineCtwo[[doubles[[tri1[[q,1]],3]]]],
nineCthree[[tri3[[i,2]]]]]]==9&&
Mod[a1+b1+c1,9]==0,
For[d=1,d≤3,d++,
For[e=1,e≤3,e++,
For[f=1,f≤3,f++,
If[Mod[a[[d]]+b[[e]]+c[[f]],9]==0,
```

```
For[d1 = 1, d1 ≤ 2, d1++,
For[e1 = 1, e1 ≤ 2, e1++,
For[f1 = 1, f1 ≤ 2, f1++,
If[Mod[a[[Mod[d + d1 - 1, 3] + 1]] + b[[Mod[e + e1 - 1, 3] + 1]] +
c[[Mod[f + f1 - 1, 3] + 1]], 9] == 0,
dblorig = Join[dblorig, {{q, p, i}}];
i = Length[tri3];
d = 3; e = 3; f = 3;
d1 = 3; e1 = 3; f1 = 3,
Null](* if statement *)
]]], Null](* if statement *)
]]], Null]]], (*for[p = 1*)
ProgressIndicator[p, {1, Length[tri2]}]
](*monitor*)
](*timing*)
](*Print*)
], ProgressIndicator[q, {1, Length[tri1]}]]
```

The above portion of the problem should probably not be allowed to run to completion - at the time (and on the computer) of writing, it takes approximately one minute to evaluate possible solutions at each of the 26046 values within **tri1**. The first 49 solutions are printed below.

dblorig[[1;;49]]

{{1,1,156},{1,5,156},{1,6,156},{1,8,54},{1,12,54},{1,14,163},{1,18,163},
{1,20,36},{1,23,36},{1,25,72},{1,28,72},{1,30,179},{1,32,179},{1,35,179},
{1,37,156},{1,40,179},{1,41,156},{1,45,156},{1,46,288},{1,48,110},{1,49,233},
{1,53,233},{1,54,63},{1,57,163},{1,59,260},{1,60,378},{1,62,360},{1,64,210},
{1,65,42},{1,66,210},{1,67,42},{1,69,72},{1,71,232},{1,72,242},{1,78,102},
{1,79,125},{1,82,288},{1,83,125},{1,84,289},{1,86,288},{1,88,110},{1,89,233},
{1,92,42},{1,94,233},{1,99,118},{1,100,233},{1,106,101},{1,112,101},{1,113,234}}

The Results

The first solution given by this program is **dblorig**[[**1**]]={**1,1,156**}. Here we translate this solution into triangles.

Input:

$q = 1$;

$p = 1$;

ii = 156;

nineCtwo[[doubles[[tri1[[q, 1]], 1]]]]

nineCtwo[[doubles[[tri1[[q, 1]], 2]]]]

nineCtwo[[doubles[[tri1[[q, 1]], 3]]]]

nineCthree[[tri1[[q, 2]]]]

Output:

{0,1}

{2,4}

{3,6}

{0,5,1}

This output tells us that the first set of 3 triangles is $\{\{0_1, 1_1, 0_2\}, \{2_1, 4_1, 5_2\}, \{3_1, 6_1, 1_2\}\}$.

Input:

nineCtwo[[**doubles**[[**dbl2**[[**tri2**[[*p*, 1]]]], 1]]]]

nineCtwo[[**doubles**[[**dbl2**[[**tri2**[[*p*, 1]]]], 2]]]]

nineCtwo[[**doubles**[[**dbl2**[[**tri2**[[*p*, 1]]]], 3]]]]

nineCthree[[**tri2**[[*p*, 2]]]]

Output:

$\{2, 3\}$

$\{4, 7\}$

$\{6, 8\}$

$\{0, 3, 1\}$

This output tells us that the second set of 3 triangles is $\{\{2_2, 3_2, 0_3\}, \{4_2, 7_2, 3_3\}, \{6_2, 8_2, 1_3\}\}$.

Input:

nineCtwo[[**doubles**[[**dbl3**[[**tri3**[[**ii**, 1]]]], 1]]]]

nineCtwo[[**doubles**[[**dbl3**[[**tri3**[[**ii**, 1]]]], 2]]]]

nineCtwo[[**doubles**[[**dbl3**[[**tri3**[[**ii**, 1]]]], 3]]]]

nineCthree[[**tri3**[[**ii**, 2]]]]

Output:

$\{2, 5\}$

$\{4, 8\}$

$\{6, 7\}$

$\{7, 8, 5\}$

This output tells us that the third set of 3 triangles is $\{\{2_3, 5_3, 7_1\}, \{4_3, 8_3, 8_1\}, \{6_3, 7_3, 5_1\}\}$.

Index