k-star Decompositions of Lambda-fold Complete Multipartite Graphs

Matthew Paul Anzur

Certificate of Approval:

C.C. Lindner
Distinguished University Professor
Mathematics and Statistics

Dean Hoffman, Chair
Professor
Mathematics and Statistics

Douglas Leonard
Professor
Mathematics and Statistics

Kevin Phelps
Professor
Mathematics and Statistics

Chris Rodger
Professor
Mathematics and Statistics

Michael Bozack
Professor
Physics

Joe F. Pittman
Interim Dean
Graduate School

k-star Decompositions of Lambda-fold Complete Multipartite Graphs

Matthew Paul Anzur

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
August 4, 2007

k-star Decompositions of Lambda-fold Complete Multipartite Graphs

Matthew Paul Anzur

Signature of Author

Date of Graduation

Vita

Matthew Paul Anzur was born October 17, 1975 in Pittsburgh, Pennsylvania. He is the son of Frank Anzur and Marcia (Przywara) Anzur. He graduated from Jacksonville High School in Jacksonville, Alabama in 1994. He entered Auburn University in August of 1994 and graduated cum laude in December of 1998 with a Bachelor of Science degree in Mathematics. He entered Graduate School, Auburn University in January of 1999. He worked as a Graduate Teaching Assistant through his time in Graduate School.

Dissertation Abstract

k-star Decompositions of Lambda-fold Complete Multipartite Graphs

Matthew Paul Anzur

Doctor of Philosophy, August 4, 2007
(B.S., Auburn University, 1998)

57 Typed Pages

Directed by Dean Hoffman

We examine the problem of k-star decompositions on λ-fold complete multipartite graphs. After a brief examination of the computational complexity issues involved, we present complete proofs for necessary and sufficient conditions in the case where $k = 2$ and the case where $\lambda = 2$ and $k = 3$. We then show some partial results for $k = 3$ and higher values of λ along with some helpful tools, including some necessary conditions, which may help in solving further cases.

Acknowledgments

The author wishes to thank the members of his advisory committee, especially Professor Dean Hoffman, for their helpful comments and suggestions. He also wishes to thank Professors Darrell Hankerson and Edward Slaminka for creating and maintaining the style files and templates used in the preparation of this dissertation. Finally, he thanks his family for all of their support.

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliography follows van Leunen's *A Handbook for Scholars*.

Computer software used The document preparation package TeX (specifically LaTeX) together with the departmental style-file `aums.sty` and text editor WinEdt. Figures were prepared in Mathematica or CorelDraw.

Table of Contents

List of Figures

Chapter 1

Introduction

An *H-decomposition* (or *H-design*) of a graph G is a set of subgraphs of G, called *blocks*, with the properties that each block is isomorphic to H and every edge in G is in exactly one block. A *Steiner Triple System*, for example, is a K_3-decomposition of K_n. We will be examining the problem of k-star decompositions of λ-fold complete multipartite graphs. First, some definitions.

1.1 Definitions

For most definitions we will use terms as defined in [1]. We will use the notation $G = (V(G), E(G))$ where $V(G)$ is the set of vertices of G and $E(G)$ is the set of edges of G. A *λ-fold complete multipartite graph* is a graph in which the vertices are partitioned into p partite sets (or *parts*) $A_1, \ldots, A_p$ with the property that each pair of vertices have either λ edges between them (if they are in different parts) or zero edges between them (if they are in the same part). We will use the notation $\lambda K_{a_1,\ldots,a_p}$ where $a_i = |A_i|$ to indicate a λ-fold complete multipartite graph. We will also put the parts in increasing order of size so $a_1 \leq a_2 \leq \ldots \leq a_p$. Then, let $n = \sum_{i=1}^{p} a_i$ be the total number of vertices in the graph, $m = \sum_{i=1}^{p-1} \sum_{j=2}^{p} a_i a_j$ the number of edges in the underlying simple graph, and $e(G) = \lambda m$ the total number of edges of G. We will also use the notation $\mu(xy)$ to indicate the number of edges between vertices x and y and for $A, B \subseteq V(G)$ let $e(AB)$ be the number of edges with one end in A and the other in B. Finally, for $S \subseteq V(G)$, let m_S be the number of edges in the underlying induced simple subgraph on S.

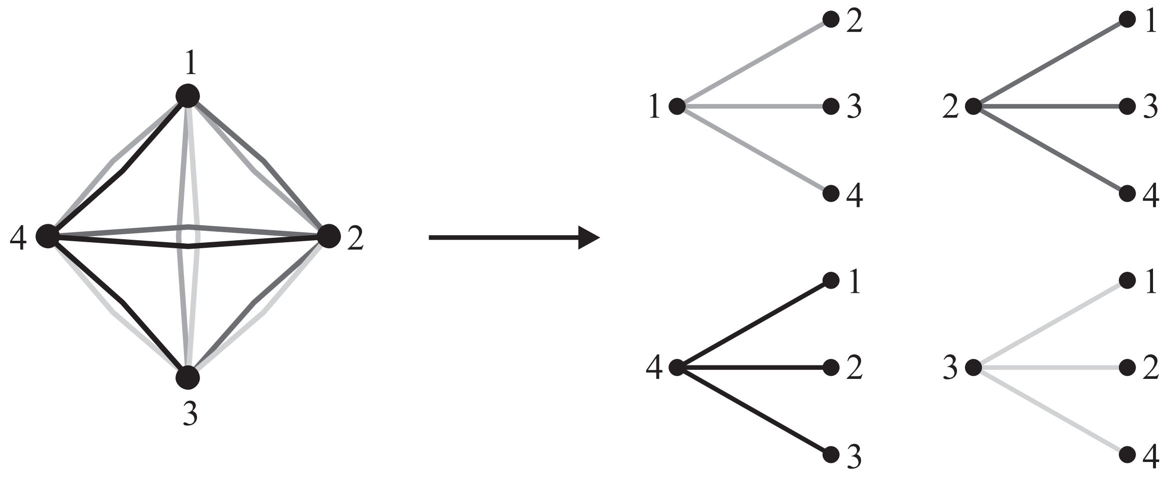

Figure 1.1: An S_3-decomposition of $2K_4$.

A *k-star* (also known as a k-claw) is a simple ($\lambda = 1$) complete multipartite graph with one part of size one and one of size k, that is, $K_{1,k}$. We will call the part of size 1 adjacent to the other k vertices the *center* of the star and the other k vertices the *ends* of the star.

1.2 A Small Example

For a small example, let us consider the graph $2K_4$. We will label the vertices as $V(G) = \{1, 2, 3, 4\}$. By simply centering one 3-star at each vertex of $2K_4$ and carefully selecting the ends, we get the S_3-design on $2K_4$ shown in Figure 1.1.

1.3 Previous Results

A *λ-fold complete graph* λK_n is a complete multipartite graph with n parts of size one and each pair of vertices has λ edges between them. In [2] Tarsi proved that the following conditions are necessary and sufficient for λK_n to have an S_k-decomposition:

Theorem 1.1 (Tarsi). *λK_n has an S_k-decomposition if and only if*

1. $\lambda n(n-1) \equiv 0 \mod 2k$

2. a. $n \geq 2k$ *if* $\lambda = 1$

   b. $n \geq k+1$ *if* λ *is even*

   c. $n \geq k+1+\frac{k}{\lambda}$ *if* λ *is odd*

In [3] Tazawa proved that the following conditions are necessary and sufficient for $K_{a_1,\ldots,a_p}$ to have an S_k-decomposition:

Theorem 1.2 (Tazawa). *$K_{a_1,\ldots,a_p}$ has an S_k-decomposition if and only if*

1. $k|m$

2. $k(n-a_p) \leq m$ *if* $k \leq n-a_p$

3. $\frac{m}{k}\big|(n-a_p)$ *if* $n-a_p < k$

With these results in mind, we will assume that G is neither simple nor K_n; that is, $\lambda \geq 2$ and $a_p \geq 2$.

1.4 $k = 2$ and Complexity

In [4], Priesler and Tarsi proved that S_k-decomposition can be done in polynomial time if $k = 2$ and it is NP-complete for multigraphs of any multiplicity λ and $k \geq 3$. Decomposing graphs into copies of S_2 is equivalent to finding a perfect matching in the λ-line graph of G, $L_\lambda(G)$. $L_\lambda(G)$ consists of a vertex for each edge of G and two vertices are adjacent in $L_\lambda(G)$ if and only if they form a path of length 2 in G. Edmonds proved in [5] that a

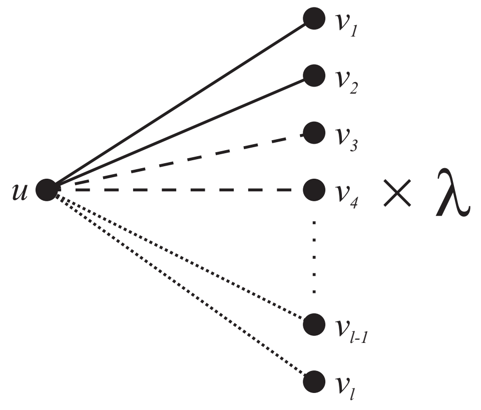

Figure 1.2: S_2-decomposition when $N(u) = 2j$.

perfect matching could be found in polynomial time. Priesler and Tarsi give a method that is easier to check in [4] for determining whether a graph admits an S_2-decomposition, but for $\lambda K_{a_1,\ldots,a_p}$ one must only check that $2|e(G)$, $n \geq 3$, and $p \geq 2$.

Theorem 1.3. *$G = \lambda K_{a_1,\ldots,a_p}$ has an S_2-decomposition if and only if $2\,|\,e(G)$, $n \geq 3$, and $p \geq 2$.*

Proof. Necessity is clear. For sufficiency, suppose first that λ is even; in this case, we will use induction. Start with a vertex $u \in A_1$. Let $\{v_1, \ldots, v_t\} = N(u)$, the vertices adjacent to u. If $t = 2j$ we may center j copies of S_2 with ends $\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{t-1}, v_t\}$ and repeat this λ times. Figure 1.2 shows an example.

If $t = 2j+1$, we may first center λj stars at u with ends $\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{t-2}, v_{t-1}\}$. Then reorient one copy of S_2 with ends $\{v_i, v_{i+1}\}$ to have its ends as $\{v_i, v_t\}$. Finally, add $\frac{\lambda}{2}$ more stars centered at u with ends $\{v_{i+1}, v_t\}$. All edges incident to u have now been covered. Repeat the same assignment in all other vertices in A_1. If there are still uncovered

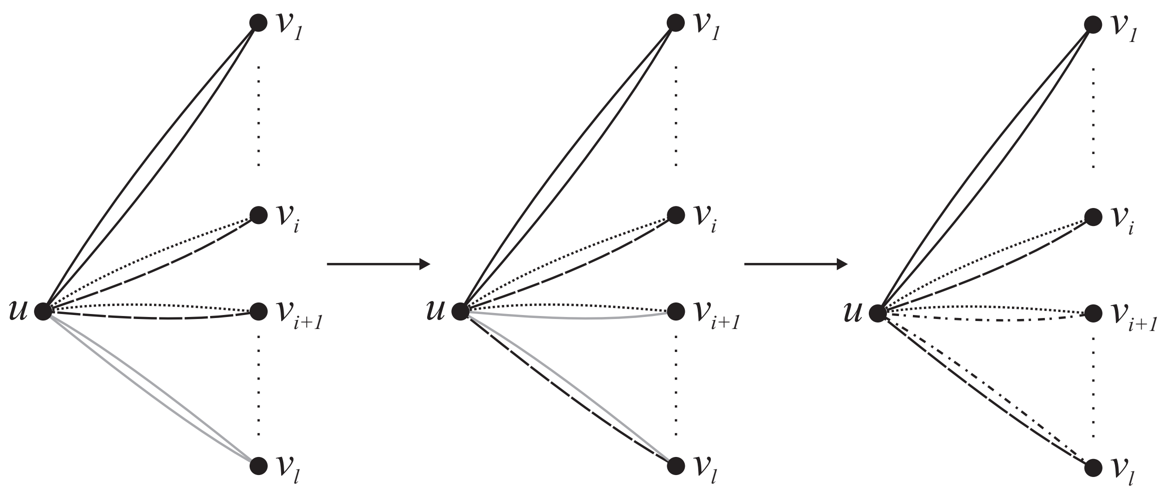

Figure 1.3: S_2-decomposition when $N(u) = 2j + 1$ and $\lambda = 2$.

edges remaining, move on to the next-indexed part and repeat the above steps until no uncovered edges remain. An example is shown in Figure 1.3.

If λ is odd, then we need $2 \mid m$. But then this (and $a_p \geq 2$ as previously assumed) is all that is needed to satisfy Tazawa's three conditions in [3] for an S_2-decomposition in the underlying simple graph. Thus we need only take an S_2-decomposition on the underlying simple graph and make λ copies of it. $\square$

Not surprisingly, it is rather easy to determine whether an S_2-decomposition exists for $\lambda K_{a_1,\ldots,a_p}$ and then construct it. In the chapters that follow, however, we take on values of $k \geq 3$; again, not surprisingly, things get much more complicated as [4] has shown that the general problem is NP-complete. For $2K_{a_1,\ldots,a_p}$ and $k = 3$, we have necessary and sufficient conditions that may be quickly checked, and present this in Chapter 3.

In Chapter 2 we will start by gathering some information and tools for narrowing the search down a bit as well as some necessary (but not always sufficient) conditions. We will then examine the case where $\lambda = 2$ and present necessary and sufficient conditions for $2K_{a_1,\ldots,a_p}$ to have an S_3-decomposition. The case where $\lambda \geq 3$ and $k = 3$, presented in Chapter 4 is unfortunately incomplete at this time. There are a few subcases where greater values of λ pull the inequalities we will be using a little too much and we cannot, at this time, draw any conclusions. We will, however, present the subcases that we have completed in Chapter 4.

Chapter 2

Our General Approach for $\lambda \geq 2$ and $k \geq 3$

Now we will begin putting together some helpful tools to help us in the search for graphs with S_k-decompositions. We will start with an important theorem giving us a place to start looking. We then present some small Lemmas useful in many of the cases to follow and necessary conditions for $\lambda K_{a_1,\ldots,a_p}$ to have an S_k-decomposition. Finally we construct a set of inequalities that will be our main tool in proving the sufficiency of these conditions.

2.1 Central Functions

We will define a *central function* on the vertices of a graph G as $c : V(G) \to \mathbb{N}$ where $\mathbb{N}$ is the set of nonnegative integers and $c(x)$ is the number of blocks of the S_k-decomposition whose center is x. In [6] Hoffman proved that the following conditions are necessary and sufficient for any graph to have an S_k-decomposition with a given function c:

Theorem 2.1 (Hoffman)**.** *G has an S_k-decomposition with central function c if and only if:*

1. $k \sum_{x \in V(G)} c(x) = e(G)$
2. $\mu(xy) \leq c(x) + c(y)$ *for all* $x, y \in \binom{V(G)}{2}$, *where* $\binom{V(G)}{2}$ *is the set of all pairs of vertices in* G
3. $e(S) + \sum_{\substack{x \in S \\ y \notin S}} \min\{c(x), \mu(xy)\} \geq kc(S)$ *for all* $S \subseteq V(G)$

We will use this theorem as the basis for finding necessary conditions specific to $\lambda K_{a_1,\ldots,a_p}$.

2.1.1 The Greedy Central Function

In most of the cases we will examine, we shall use a greedy central function to attempt to center copies of S_k in G. Let C_i, the *capacity* of the part A_i, be the maximum number of blocks of an S_k-decomposition we may center at a vertex in A_i. If $n - a_i \geq k$, then $C_i = \left\lfloor \frac{\lambda(n-a_i)}{k} \right\rfloor$; if $n - a_i < k$ then $C_i = 0$. We will say that a part A_i is *at capacity* if $c(t) = C_i$ for all $t \in A_i$. We will run the greedy central function algorithm as follows:

Initialization: Let $i = 1$, $c(t) = 0$ for all $t \in V(G)$, and $r = e(G)/k$.

1. If A_i is not already at capacity:

   a. If $r \geq a_i$ increase $c(t)$ by one at each vertex $t \in A_i$ and decrease r by a_i. If $i = p$, set $i = 1$ and repeat Step 1. If $i < p$, increase i by one and repeat Step 1.

   b. If $0 < r < a_i$, let $t_1, \ldots, t_{a_i}$ be the vertices in A_i. Increase $c(x)$ by one on vertices $t_1, \ldots, t_r$ and stop.

   c. If $r = 0$, stop.

2. If A_i is at capacity, set $i = 1$ and go back to Step 1.

As the greedy central function respects the capacities of each part and does not move up to a higher value until the all vertices not already at capacity have been increased, there will be at most one part A_β with $c(t) = c(u) + 1$ for some vertices t and u in A_β. We will call A_β the *split part*. In every other part, c is constant; for simplicity, we will let q_i be the least value of $c(x)$ for $x \in A_i$, $T_i \subseteq S$ be the vertices x in non-split parts with $c(x) = i$, and $t_i = |T_i|$.

As we will see in Chapters 3 and 4, in most cases where $k = 3$ this greedy central function will yield an S_3-decomposition of $\lambda K_{a_1,\dots,a_p}$. There are a few exceptions when $\lambda = 2$, but we have backup central functions that work. There are, however, possibly several cases when $\lambda \geq 3$ where this function will not yield an S_3-decomposition.

2.2 Some General Results and Lemmas

We have a few more or less straightforward (but helpful) lemmas to aid us in the cases ahead. This result for graphs where the greedy central function stops on a split part is trivial, but as we refer to it in several cases that follow, we present it as a lemma.

Lemma 2.1. *If the greedy central function c terminates in A_β and A_β is a split part, then $\max_{x \in V(G)} \{c(x)\} = q_\beta + 1$. If A_β is not split, then $\max_{x \in V(G)} \{c(x)\} = q_\beta$.*

Proof. As c respects the capacities of each part, the central function algorithm will only go back to A_1 (and thus increase the maximum value of c) when it has added one to the value of c in all of the vertices in parts that are not already at capacity. Thus, if the algorithm ends in a part A_β, then the central function value in all parts A_i where $i < \beta$ must be the same, namely, the greatest value of $c(x)$ for $x \in A_\beta$. This is $q_\beta + 1$ if A_β is split, and q_β if it is not. As the central function is decreasing as the indices increase, this is the maximum value of c. $\square$

The following is simply a straightforward counting of edges in a subset $S \subseteq V(G)$, but again, it is used in many cases ahead so we present it as a lemma.

Lemma 2.2. *If $S \subseteq V(G)$ consists of p parts of size at least t, then $m_S \geq \frac{p-1}{p} st$, where $s = |S|$.*

Proof. If S consists of p parts of size at least t, then

$$\frac{s}{p}t \geq \left(\frac{pt}{p}\right) t = t^2$$

So then

$$m_S \geq t(s-t) = st - t^2 \geq st - \frac{st}{p} = \left(\frac{p-1}{p}\right)$$

□

2.2.1 Some Necessary Conditions

Here we present two necessary conditions for a graph to have an S_k-decomposition. In the case where $\lambda = 2$ and $k = 3$, as we will see in Chapter 3 these are also sufficient. For $k = 3$ and higher values of λ, we will need another condition which we present in Chapter 4.

Theorem 2.2. *If $G = \lambda K_{a_1,\ldots,a_p}$ has an S_k-decomposition, then*

1. $k \,|\, e(G)$
2. a. $k(n - a_p) \leq m$ *if* $2a_p \geq n$

   b. $n + \frac{\varepsilon}{\lambda}(n - 2a_p) \leq \frac{2m}{k}$ *where* $\varepsilon = \lambda \bmod 2$ *if* $2a_p \leq n$.

Proof. The necessity of Condition 1 is obvious; we cannot have any leftover edges. Now we examine each case of Condition 2.

Case 1 $(2a_p \geq n)$ Condition 2a. is equivalent to $\frac{\lambda m}{k} \geq \lambda(n - a_p)$. Suppose then, that $\frac{\lambda m}{k} < \lambda(n - a_p) \leq \frac{\lambda n}{2}$. There must be, then, a vertex x in some part A_i with $q = c(x) < \frac{\lambda}{2}$. Hoffman's Condition 2 in Theorem 2.1 must hold here; thus every other vertex y in $G \setminus A_i$ must have $c(y) \geq \lambda - q$. But then

$$\begin{aligned}\lambda(n - a_p) > \frac{\lambda m}{k} &\geq (\lambda - q)n - (\lambda - q)a_i + qa_i \\ &= (\lambda - q)n - (\lambda - 2q)a_i \\ &\geq (\lambda - q)n - (\lambda - 2q)a_p \\ &= \lambda(n - a_p) + q(2a_p - n)\end{aligned}$$

But here $2a_p \geq n$, so the above inequality cannot hold.

Case 2 $(2a_p \leq n)$ We will examine subcases by whether λ is even or odd.

Case 2.1 (λ is even) Here Condition 2b. is equivalent to $\frac{\lambda m}{k} \geq \frac{\lambda n}{2}$. Suppose, then, that $\frac{\lambda m}{k} < \frac{\lambda n}{2}$. There must be, then, a vertex x in some part A_i with $q = c(x) < \frac{\lambda}{2}$. Again by Hoffman's Condition 2 in Theorem 2.1 we need that for every other vertex y in $G \setminus A_i$ that

$c(y) \geq \lambda - q$. But then we need

$$\begin{aligned}
\frac{\lambda n}{2} > \frac{\lambda m}{k} &\geq (\lambda - q)(n - a_i) + q a_i \\
&= (\lambda - q)n - (\lambda - 2q)a_i \\
&\geq (\lambda - q)n - (\lambda - 2q)a_p \\
&= (\lambda - q)(n - a_p) + q a_p \\
&= \left(\lambda - q - \frac{\lambda}{2}\right)(n - a_p) + \frac{\lambda}{2}(n - a_p) + q a_p \\
&= \frac{\lambda n}{2} + \left(\frac{\lambda}{2} - q\right)(n - 2a_p)
\end{aligned}$$

But with $q < \frac{\lambda}{2}$ and $n \geq 2a_p$, the above inequality cannot hold.

Case 2.2 (λ is odd) Here Condition 2b. is equivalent to $\frac{\lambda m}{k} \geq \frac{\lambda+1}{2}n - a_p$. Suppose, then, that $\frac{\lambda m}{k} < \frac{\lambda+1}{2}n - a_p$. There must be, then, a vertex x in some part A_i with $q = c(x) \leq \frac{\lambda-1}{2}$. Once again by Hoffman's Condition 2 in Theorem 2.1 we need that for every other vertex y in $G \setminus A_i$ that $c(y) \geq \lambda - q$. We have, then, that

$$\begin{aligned}
\frac{\lambda+1}{2}n - a_p > \frac{\lambda m}{k} &\geq (\lambda - q)(n - a_i) + q a_i \\
&= (\lambda - q)n - (\lambda - 2q)a_i \\
&\geq (\lambda - q)n - (\lambda - 2q)a_p \\
&= (\lambda - q)(n - a_p) + q a_p \\
&= \left(\lambda - q - \frac{\lambda+1}{2}\right)(n - a_p) + \frac{\lambda+1}{2}(n - a_p) + q a_p \\
&= \frac{\lambda+1}{2}n - a_p + \left(\frac{\lambda-1}{2} - q\right)(n - 2a_p)
\end{aligned}$$

But with $q \leq \frac{\lambda-1}{2}$ and $n \geq 2a_p$, the above inequality cannot hold.

$\square$

2.3 The Subsets and The Inequalities

In most cases of the proofs for sufficiency of these conditions, we will show that the greedy central function meets Hoffman's Condition 3 in Theorem 2.1. Checking every possible subset of $V(G)$, however, is a daunting task; we want to narrow down the subsets $S \subseteq V(G)$ that we need to examine. From this point forward, we will let $S \subseteq V(G)$ be a largest subset that minimizes

$$f(S) = e(S) - 3c(S) + \sum_{\substack{x \in S \\ y \notin S}} \min\{c(x), \mu(xy)\} \tag{2.1}$$

The following lemma shows that, in each part, all of the vertices with the same value of c are all either in S or none of them are. Thus, if the central function is constant across each part (and the only one where it may not be is the part where the algorithm stops), then we may select S part-by-part instead of vertex-by-vertex. If a part A_i is split, either all of the vertices are in S, none of them are, only those with $c(x) = q_i$ are, or only those with $c(x) = q_i + 1$ are.

Lemma 2.3. *Let x and y be two vertices in a part A_i such that $c(x) = c(y)$. Then $x \in S$ if and only if $y \in S$.*

Proof. Let $q = c(x) = c(y)$. Assume without loss of generality that $x \in S$ and $y \notin S$. Removing x from S increases f by

$$-\lambda(n-a_i) + kq - \sum_{t\notin S} \min\{q, \mu(xt)\} + \sum_{t\in S} \min\{c(t), \mu(xt)\} \geq 0$$

However, all terms in the above sums of minimums corresponding to vertices in A_i other than x are zero, as there are no edges between them. We may ignore those terms. So the above inequality becomes

$$-\lambda(n-a_i) + kq - \sum_{t\notin S\cup A_i} \min\{q, \lambda\} + \sum_{\substack{t\in S\\ t\notin A_i}} \min\{c(t), \lambda\} \geq 0 \tag{2.2}$$

Adding y to S increases f by

$$\lambda(n-a_i) - kq + \sum_{t\notin S} \min\{q, \mu(yt)\} - \sum_{t\in S} \min\{c(t), \mu(yt)\} > 0$$

As above, we can ignore all terms in the sums corresponding to vertices in A_i other than y. The above inequality becomes

$$\lambda(n-a_i) - kq + \sum_{t\notin S\cup A_i} \min\{q, \lambda\} - \sum_{\substack{t\in S\\ t\notin A_i}} \min\{c(t), \lambda\} > 0 \tag{2.3}$$

Adding the corresponding left and right sides of 2.2 and 2.3 above gives $0 > 0$, obviously a contradiction. $\square$

With this lemma in mind, we will define the following subsets of indices of the p parts. Let $I = \{1, 2, \ldots, p\}$ be the indices of the parts, $B_i = \{x \in A_i \,|\, c(x) = q_i\}$ be the vertices in a part with the least value of the central function, and $b_i = |B_i|$. Then let

$$\begin{aligned}
U &= \{i \in I | B_i = A_i\} \\
\overline{U} &= \{i \in I | b_i < a_i\} \\
V &= \{i \in U | A_i \cap S = \varnothing\} \\
W &= \{i \in U | A_i \subseteq S\} \\
X &= \left\{i \in \overline{U} | A_i \cap S = \varnothing\right\} \\
Y &= \left\{i \in \overline{U} | A_i \cap S = B_i\right\} \\
Y' &= \left\{i \in \overline{U} | A_i \cap S = A_i \backslash B_i\right\} \\
Z &= \left\{i \in \overline{U} | A_i \subseteq S\right\}
\end{aligned}$$

Now instead of picking subsets vertex-by-vertex, we may select them part-by-part (or one of the "halves" of the part if it is split). We still have a lot of work to do, and there are still many possible subsets, but this greatly simplifies things.

2.3.1 The Inequalities

Now we may put together the primary means of proving the cases to come. The following inequalities result from giving a largest set minimizing S a small nudge; that is, we will either add one part to S that was not in there before or remove one part from S and examine the change in f. As we are assuming that S is a largest set that minimizes f, adding any vertices to S will always increase f; removing vertices may or may not increase f but will not decrease it.

For all $v \in V$, adding A_v to S gives

$$\lambda s a_v - k q_v a_v + (n - s - a_v) \sum_{t \in A_v} \min\{q_v, \lambda\} - a_v \sum_{t \in S} \min\{c(t), \lambda\} > 0 \tag{2.4}$$

For all $w \in W$, removing A_w from S gives

$$-\lambda(s - a_w) a_w + k q_w a_w - (n - s) \sum_{t \in A_w} \min\{q_v, \lambda\} + a_w \sum_{t \in S \backslash A_w} \min\{c(t), \lambda\} \geq 0 \tag{2.5}$$

For all $x \in X$, adding B_x to S gives

$$\lambda s b_x - k q_x b_x + (n - s - a_x) \sum_{t \in B_\alpha} \min\{q_x, \lambda\} - b_x \sum_{t \in S} \min\{c(t), \lambda\} > 0 \tag{2.6}$$

Adding $A_x \backslash B_x$ to S gives

$$\lambda s(a_x - b_x) - k(q_x + 1)(a_x - b_x) + (n - s - a_x) \sum_{t \in Ax \backslash B_x} \min\{q_x, \lambda\} - (a_x - b_x) \sum_{t \in S} \min\{c(t), \lambda\} > 0 \tag{2.7}$$

For all $y \in Y$, adding B_y to S gives

$$\lambda s b_y - k q_y b_y + (n - s - b_y) \sum_{t \in B_y} \min\{q_y, \lambda\} - b_y \sum_{t \in S \backslash (A_y \backslash B_y)} \min\{c(t), \lambda\} > 0 \tag{2.8}$$

Removing $A_y \setminus B_y$ from S gives

$$\begin{aligned} &- \lambda(s - (a_y - by))(a_y - b_y) + k(q_y + 1)(a_y - b_y) - \\ &\qquad (n - s - b_y) \sum_{t \in A_y \backslash B_y} \min\{q_x, \lambda\} + (a_y - b_y) \sum_{t \in S \backslash (A_y \backslash B_y)} \min\{c(t), \lambda\} \geq 0 \end{aligned} \tag{2.9}$$

For all $y \in Y'$, removing B_y from S gives

$$-\lambda(s-b_y)b_y + kq_yb_y - (n-s-(a_y-b_y))\sum_{t\in B_y}\min\{q_y,\lambda\} + b_y\sum_{t\in S\setminus B_y}\min\{c(t),\lambda\} \geq 0 \quad (2.10)$$

Adding $A_y \setminus B_y$ to S gives

$$\begin{aligned}&\lambda s(a_y-b_y) - k(q_y+1)(a_y-b_y) + \\ &\qquad (n-s-(a_y-b_y))\sum_{t\in A_y\setminus B_y}\min\{q_y,\lambda\} - (a_y-b_y)\sum_{t\in S\setminus B_y}\min\{c(t),\lambda\} > 0 \quad (2.11)\end{aligned}$$

For all $z \in Z$, removing B_z from S gives

$$-\lambda(s-a_z)b_z + kq_zb_z - (n-s)\sum_{t\in B_x}\min\{q_z,\lambda\} + b_x\sum_{t\in S\setminus A_z}\min\{c(t),\lambda\} \geq 0 \quad (2.12)$$

Removing $A_z \setminus B_z$ from S gives

$$\begin{aligned}&-\lambda(s-(a_z-b_z))(a_z-b_z) + k(q_z+1)(a_z-b_z) - \\ &\qquad (n-s)\sum_{t\in A_z\setminus B_z}\min\{q_z,\lambda\} + (a_z-b_z)\sum_{t\in S\setminus(A_z\setminus B_z)}\min\{c(t),\lambda\} \geq 0 \quad (2.13)\end{aligned}$$

As awful as some of these inequalities look, they are very valuable tools in examining sufficiency in the cases to come. We will generally divide these cases by the value of $\min\{q_i,\lambda\}$, greatly simplifying the work ahead. In fact, if $\min\{q_i,\lambda\} = \lambda$ we usually only have two terms in each inequality to deal with. Now, we will put them to use in the case where $k = 3$ and $\lambda = 2$.

Chapter 3

$k = 3, \lambda = 2$

Here we present our result on S_3-decompositions of 2-fold complete multipartite graphs. We will show that conditions given in Theorem 2.2 are necessary and sufficient.

Theorem 3.1. *Let $G = 2K_{a_1,\ldots a_p}$. G has an S_3-decomposition if and only if:*

1. $3 \mid m$
2. a. *If $2a_p \leq n$ then $3n \leq 2m$*

   b. *If $2a_p \geq n$ then $3(n - a_p) \leq m$*

Proof. The necessity of these conditions was proved in Theorem 2.2. For sufficiency, we will examine four main cases. In the first three we will assume that the capacities of each part are positive and then examine cases according to the values c takes, specifically, when $c(x)$ is at most 2 or at least 2 for all $x \in V(G)$. For the last case, we will examine graphs where the capacity of A_p is zero.

We will let $S \subseteq V(G)$ be a largest subset that minimizes

$$f(S) = e(S) - 3c(S) + \sum_{\substack{x \in S \\ y \notin S}} \min\{c(x), \mu(xy)\}$$

Note that $f(V(G)) = f(\varnothing) = 0$; we will assume that S is neither. We then use the inequalities and subsets from Section 2.3 to find all of the possible subsets S that minimize $f(S)$. In each of these cases, our goal will either be to show that even if that set minimizes

S, $f(S) \geq 0$, or, that S cannot minimize f due to some contradiction. We begin with the case where all vertices receive exactly one star from the greedy central function.

Case 1 ($q_i = 1$ for all i): We may assume that $V \neq \varnothing \neq W$, or else we would have $S = G$ or $S = \varnothing$. From 2.4 and 2.5 we get $a_w > a_v$. Here $f(S) = e(S) - s(3 - (n - s))$; with $a_w \geq 2$ by Lemma 2.2, $e(S) = 2m \geq 2s$, so $f(S) \geq 0$.

Case 2 ($1 \leq q_i \leq 2$, $q_i = 1$ for some $i \in I$, $q_j = 2$ for some $j \in I$)

Case 2.1 ($V \neq \varnothing \neq W$): From 2.4 and 2.5 we have

$$[3 - (n - s)](q_w - q_v) + (2 - q_w)a_w - a_v q_v - 1 \geq 0 \tag{3.1}$$

so we cannot have $q_v = q_w = 2$ for any $v \in V$ or $w \in W$; in other words, the partite sets with $q_i = 2$ are all either in V or W.

Case 2.1.1 ($\{\, i \,|\, q_i = 2\} \subseteq W$) Here $q_v = 1$ for all $v \in V$. Then the inequality in 3.1 becomes $[3 - (n - s)] \geq a_v + 1$ so we must have $a_v = n - s = 1$ and thus $\overline{U} = \varnothing$; the central function must be constant across each part. Thus S consists of all of the vertices in G except for a part A_β of size 1 with $a_\beta = 1$. So then

$$\begin{aligned} f(S) &= e(S) - 2c(S) \\ &= 3c(G) - 2(n-1) - 2[c(G) - 1] \\ &= c(G) - 2n + 4 \\ &= 2n - 1 - s_1 - 2n + 4 \\ &= 3 - s_1 \end{aligned}$$

We will then assume that $s_1 > 3$ and eliminate this case by contradiction.

With $G \neq 2K_n$, there must be a $w \in W$ such that $a_w \geq 2$ and $q_w = 1$. Here, then, $T_j = \bigcup_{\substack{i \in W \\ q_i = j}} A_i$ and $t_j = |T_j|$. Let A_{w_o} be a smallest part in T_1. Removing A_{w_0} from S we get from 2.5

$$-2(n-1-a_{w_0}) + 2 + (c(S) - a_{w_0}) \geq 0$$

and by our assumptions in this case, $c(S) = 2n - 2 - s_1$ so the above inequality becomes $s_1 - a_{w_0} \leq 2$; removing A_{w_1} would leave at most 2 vertices in T_1. And, by the inequality in 3.1 above, $a_{w_o} > a_v = 1$. With $c(G) = 2n - 1 - s_1$, $m = 3n - \frac{3}{2}(s_1 + 1)$ so s_1 must be odd. But then if removing the least-sized part in T_1 leaves at most two vertices in T_1 and $A_p \subseteq T_1$ has size at least 2, we must have $T_1 = A_p$ and so G consists of $p-1$ parts of size 1 and A_p.

We will now count edges to arrive at the promised contradiction. We must have here that

$$\begin{aligned} e(G) = 6n - 3a_p - 3 &= 2\binom{n-a_p}{2} + 2a_p(n-a_p) \\ &= (n-a_p)(n-a_p-1) + 2a_p(n-a_p) \\ &= n^2 - n + a_p - a_p^2 \end{aligned}$$

This is a little easier to prove if we substitute $n = t_2 + 1 + a_p$. With a little shuffling of terms, the above equation becomes

$$a_p + 3 = t_2(t_2 - 5) + 2a_p t_2 \geq -6 + 4a_p$$

But this is only true when $a_p \leq 3$, contradicting our assumption here that $s_1 > 3$.

Case 2.1.2 $(\{i \,|\, q_i = 2\} \subseteq V)$

Case 2.1.2.1 ($\overline{U} = \varnothing$) Here $c(S) = s$ and $f(S) = e(S) - [3-(n-s)]s$. By 3.1, for $a_{w_1} \in W$ and $v_2 \in V$ with $q_{v_2} = 2$, we have

$$a_{w_1} \geq [3-(n-s)] + 2a_v + 1$$

S must consist of at least two parts, as the greedy central function respects the capacities of each part, all parts have capacity of at least 1, and $q_i \geq 1$ for all i by assumption. Let A_{w_0} be a smallest part in T_1. By the above inequality, $a_{w_1} \geq 4$ and then by Lemma 2.2, $e(S) \geq 4s$ and so $f(S) \geq 0$.

Case 2.1.2.2 ($\overline{U} \neq \varnothing$) If $\overline{U} = X$, $f(S) = e(S) - [3-(n-s)]s$. With $V \neq \varnothing$ and $a_x \geq 2$ we have $n - s \geq 3$ and so $f(S) \geq 0$. If $\overline{U} = Y$, we have

$$\begin{aligned} f(S) &= e(S) - 3c(S) + (n-s)c(S) - 2b_y(a_y - b_y) \\ &= e(T_1) + 2(a_y - b_y)t_1 - 2b_y(a_y - b_y) - [3-(n-s)]c(S) \\ &= 2(a_y - b_y)[t_1 - b_y] + e(T_1) - [3-(n-s)]c(S) \end{aligned}$$

and with $t_1 > b_y$, we may again assume that $n - s \leq 2$. But then this means $a_v = 1$, $b_y = 1$ (so $n - s = 2$), and $\{i \in V | q_i = 1\} = \varnothing$. So $c(G) = 2n - 1 - t_1$ and so $m = 3n - \frac{3}{2} - \frac{3}{2}s_1$; s_1, then, must be odd. By 3.1, we have for any $w_1 \in T_1$ that $a_{w_1} \geq 4$. Here, $c(S) = 2(a_y - 1) + t_1$, so we will show $2(a_y-1)[t_1 - 1] \geq 2(a_y - 1) + t_1$, or $2(a_y - 1)[t_1 - 2] \geq t_1$. But when $t_1 \geq 4$, we have $2(s_1 - 2) \geq s_1$, and so $f(S) \geq 0$.

If $\overline{U} = Y'$, then we have

$$f(S) = e(S) - [3 - (n - s)]s - b_y(a_y - b_y)$$
$$= e(T_1) + b_y[t_1 - (a_y - b_y)] - [3 - (n - s)]s$$

As $t_1 > (a_y - b_y)$, we may again assume that $n - s = 2$. But then the inequality from 2.10 becomes $t_1 \leq 2$, contradicting 3.1 above.

Lastly, in the case where $\overline{U} = Z$, we have

$$f(S) = e(S) - [3 - (n - s)]c(S) = e(S) - [3 - (n - s)](s + (a_z - b_z))$$

so we may again assume that $n - s < 3$. We will show that $m_S \geq (s + (a_z - b_z))$. In this case $c(G) = 2n - b_z - t_1$ and $m = 3n - \frac{3}{2}b_z - \frac{3}{2}t_1$ so $b_z + t_1$ must be even. If $a_z = 2$ then $b_z = 1$ and thus (as $b_z + t_1$ must be even) $s_1 \geq 3$. We then need $m_S \geq s + 1$; with $m_S \geq 2t_1$ and $t_1 \geq 3$ this is always the case. If $a_z \geq 3$, then by Lemma 2.2, $m_S \geq \frac{3}{2}s \geq s + (a_z - b_z)$, as $a_z < t_1$.

Case 2.2 $(V \neq \varnothing, W = \varnothing)$ If $\overline{U} = \varnothing$ or $\overline{U} = X$ then $S = \varnothing$ and $f(S) = 0$. So first suppose $\overline{U} = Y$. Then $f(S) = -3c(S) + (n - s - b_y)c(S)$. But G must have at least one part other than A_y and it must be in V. With $q_y = 1$ there must be at least three other vertices outside of A_y as the greedy central function respects the capacities of each part, thus $f(S) \geq 0$. The cases where $\overline{U} = Y'$ and $\overline{U} = Z$ are similar.

Case 2.3 $(V = \varnothing, W \neq \varnothing)$ If either $\overline{U} = \varnothing$ or $\overline{U} = Z$, then $S = V(G)$ and $f(S) = 0$. We will examine the remaining cases separately.

Case 2.3.1 ($\overline{U} = X$) Here $f(S) = e(S) - 3c(S) + a_x c(S)$, so we will assume that $a_x = 2$. Combining inequalities from 2.5 and 2.6 for W and X, we get $(2 - q_w)a_w + q_w - 3 > 0$, so $q_w = 1$ for all $w \in W$ thus $c(S) = s$ and $f(S) = e(S) - s$. Since $a_w \geq a_x = 2$, by Lemma 2.2 we have $e(S) \geq 2s > c(S)$.

Case 2.3.2 ($\overline{U} = Y$) In all of the other cases we have examined, we have been able to show that either $f(S) \geq 0$ directly, or, assuming that a particular set minimized f lead to a contradiction. In this case, assuming that $S = V(G) \setminus B_y$ minimizes f, this implies that $f(S) < 0$. Thus we would hope that the usual inequalities and other properties of the graph would lead to a contradiction. We will be able to use the usual inequalities to greatly narrow the field, but, as we will see, one graph will remain.

In graphs where $S = G \setminus B_y$, $f(S) = e(S) - c(S) + (n - s - b_y)c(S)$ and $c(S) = 2n - 2b_y - t_1$. Note also here that $c(G) = 2n - b_y - t_1$ and so $m = 3n - \frac{3}{2}b_y - \frac{3}{2}t_1$; thus $b_y + t_1$ must be even.

From the inequalities in 2.8 and 2.9 we have

$$2(n - a_y) - 3 - [2n - 2b_y - t_1 - 2(a_y - b_y)] > 0$$

(so $t_1 \geq 4$ and thus $T_1 \neq \varnothing$) and $-2(n - a_y) + 6 + 2n + 2a_y - t_1 \geq 0$, (and so $t_1 \leq 6$). Removing any $A_{w_1} \subseteq T_1$, from 2.5 we get $t_1 - a_w \leq 3 - b_y$. This means there are at most two more vertices in T_1 outside of A_{w_1}; so either A_p is the only part in W with $q_w = 1$ or $T_1 = A_{p-1} \cup A_p$ where $a_{p-1} = a_p = 2$. In the latter case, as $a_y \leq a_{p-1}$ we must have $a_y = 2$ and so $b_y = 1$; but this contradicts the condition above that $b_y + t_1$ must be even. So $T_1 = A_p$.

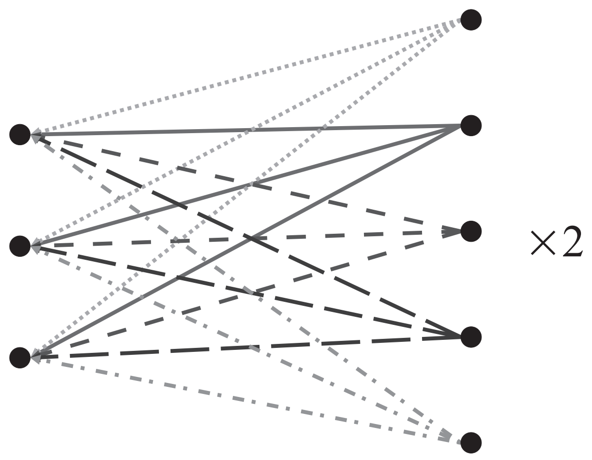

Figure 3.1: An S_3-decomposition of $2K_{3,5}$.

Now we need to see if $T_2 \neq \varnothing$. By our assumptions in this case about the central function, we have $m = 3n - \frac{3}{2}b_y - \frac{3}{2}t_1$; counting edges we have

$$\begin{aligned}
3n - \frac{3}{2}b_y - \frac{3}{2}a_p &\geq t_2(a_y + a_p) + a_y a_p \\
3t_2 + 3a_y + 3a_p - \frac{3}{2}b_y - \frac{3}{2}a_p &\geq t_2 a_y + t_2 a_p + a_y a_p \\
a_p\left(\frac{3}{2} - t_2 - a_y\right) &\geq -3t_2 + t_2 a_y + \frac{3}{2}b_y \\
6\left(\frac{3}{2} - t_2 - a_y\right) &\geq -3t_2 + t_2 a_y + \frac{3}{2}b_y \\
9 &\geq 3t_2 + (t_2 + 6)a_y + \frac{3}{2}b_y
\end{aligned}$$

Assuming that $t_2 \geq 1$, we have a contradiction. Thus, $T_2 = \varnothing$ and G must be bipartite. Here is where we have a problem; the graph $2K_{3,5}$ meets all of the conditions above (and is the only graph that does), and if we apply the greedy central function to this graph, $f(S) < 0$! Fortunately, this is the only such graph and it has a straightforward S_3-decomposition.

Letting $c(x) = 0$ for $x \in A_1$ and $c(x) = 2$ for $x \in A_2$ gives an S_3-decomposition on $2K_{3,5}$, as illustrated in Figure 3.1.

Case 2.3.3 ($\overline{U} = Y'$) Here $f(S) = e(S) - 3c(S) + c(S)[n - s - (a_y - b_y)]$. The inequalities from 2.11 and 2.10 for Y' give $2(n-a_y)-6-c(S)+b_y > 0$ and $-2(n-a_y)+3+c(S)-b_y \geq 0$; adding the left sides of these, we have $-3 > 0$, obviously a contradiction.

Case 3 ($q_i \geq 2$ for all $i \in I$) First, we will need a lemma. We will show that the greedy central function will not fill the largest two parts (A_{p-1} and A_p) to capacity. This will reduce the number of distinct values of c to at most three.

Lemma 3.1. $q_{p-1} < \left\lfloor \dfrac{2(n-a_{p-1})}{3} \right\rfloor$.

Proof. If A_{p-1} and A_p are both at capacity, then we must have

$$(n-a_p)\left\lfloor \frac{2(n-a_{p-1})}{3} \right\rfloor + a_p \left\lfloor \frac{2(n-a_p)}{3} \right\rfloor \leq \frac{2m}{3}$$

so

$$(n-a_p)\left[\frac{2(n-a_{p-1})}{3} - \frac{2}{3}\right] + a_p \left[\frac{2(n-a_p)}{3} - \frac{2}{3}\right] \leq \frac{2m}{3}$$

Maximizing the possible number of edges in $V(G)\backslash A_p$ in the graph gives

$$(n-a_p)[2(n-a_{p-1})-2] + a_p[2(n-a_p)-2] \leq 2\binom{n-a_p}{2} + 2a_p(n-a_p)$$

and so

$$(n-a_p)[n-2a_{p-1}+a_p-1] \leq 2a_p$$

G cannot be bipartite; both parts cannot be at capacity, for we would need $2m \geq 4m - 4$ which certainly cannot be the case if $3|m$. So $p \geq 3$ and we have

$$(n - a_p)\left[\sum_{i=1}^{p-2} a_i - 1 + a_p - a_{p-1} + a_p\right] \leq 2a_p$$

but with $n - a_p \geq 3$, $\sum_{i=1}^{p-2} a_i \geq 1$, and $a_{p-1} \leq a_p$, we have a contradiction. □

From this, we now know that the only step of size two or greater can be from A_{p-1} to $A_{p.}$ Note that in the case $\overline{U} = Y'$, the inequalities from 2.11 and 2.10 give a direct contradiction.

Case 3.1 ($V \neq \varnothing \neq W$) From the inequalities for V and W from 2.4 and 2.5, we get $q_w - q_v \geq \frac{2}{3}a_v + \frac{1}{3}$. (Note that this also eliminates the case where the central function is constant and $q \geq 2$.) The central function is decreasing, so we get $a_w \leq a_v$. If $q_w - q_v = 1$, we must have $a_v = 1$ for all $v \in V$, but then $a_p = 1$ and so $G = 2K_n$, which we are assuming it is not in light of Theorem 1.1. So under our previous assumption that $a_p \geq 2$ we must have $q_w - q_v \geq 2$, and A_p must be at capacity.

Case 3.1.1 ($\overline{U} = \varnothing$) Here $W = \{1, \ldots, p-1\}$, $V = p$, and $f(S) = e(G) - 3c(S) > 0$. Note that if $\overline{U} = Z$, f has the same value.

Case 3.1.2 ($\overline{U} = X$) From 2.5 and 2.6 we have $q_w - q_x \geq \frac{2}{3}a_x + \frac{1}{3}$. Since $a_x \geq 2$, $q_w - q_x \geq 2$, but we must have $\max_{i \in I}\{q_i\} = q_x + 1$ by Lemma 2.1, so we have a contradiction.

Case 3.1.3 ($\overline{U} = Y$) From inequalities 2.5 and 2.8 we have $q_w - q_y \geq \frac{2}{3}b_y + \frac{1}{3}$. Thus if $b_y \geq 2$, $q_w - q_y \geq 2$ and, similar to the previous case we have $\max_{i \in I}\{q_i\} = q_y + 1$, a contradiction. If $b_y = 1$ then we need to examine $f(S)$ a little differently. As a split part must have size at least two and $q_w > q_y \geq q_v$, all parts in V must be at capacity; by Lemma 3.1 only A_p

may be at capacity, thus $V = \{p\}$. Then

$$\begin{aligned} f(S) &= e(S) - 3c(S) + 2(s - (a_y - b_y))(n - s) + 2(a_y - b_y)a_p \\ &= e(S) - 3c(S) + e(S\overline{S}) \\ &= e(G) - e(\overline{S}) - 3c(S) \\ &= 3c(\overline{S}) - e(\overline{S}) \\ &= 3c(\overline{S}) - 2a_p > 4a_p \end{aligned}$$

and as $q_p \geq 2 = \lambda$ by assumption in this case, $f(S) > 0$.

Case 3.2 ($W = \varnothing$) First, assume that $\overline{U} = Y$. Then $f(S) = -3(q_y + 1)(a_y - b_y) + 2(a_y - b_y)(n - a_y)$, so we need $2(n - a_y) \geq 3(q_y + 1)$ or $q_y \leq \frac{2(n-a_y)}{3} - 1$. But since A_y is split, it cannot be completely filed to capacity, so $q_y \leq \left\lfloor \frac{2(n-a_y)}{3} \right\rfloor - 1$ and so $f(S) > 0$. If $\overline{U} = Z$, then from 2.12 we have $2(n - a_z) \leq 3q_z$, but this would put $q_z + 1$ over capacity.

Case 3.3 ($V = \varnothing$) Here we can only have $\overline{U} = X$ or $\overline{U} = Y$, and these cases are similar in the case where $V \neq \varnothing \neq W$ above.

Case 4 ($n - a_p < 3$ so $q_p = C_p = 0$) In this case we will use a different central function than the other cases. First, we will eliminate the case when $n - a_p < 3$ and $2a_p < n$. Our assumption that $a_p \geq 2$ leaves only three possible graphs: $2K_{1,2}$, $2K_{1,1,2}$, and $2K_{2,2}$. None of these graphs satisfy the edge-multiplicity Condition 1. Thus if $n - a_p < 3$, we must have $a_p \geq 3$. Fortunately, there are only three cases to consider here, and all three are straightforward assuming the graph satisfies $3 \mid e(G)$. If G is bipartite, then condition 1 says we must have $3 \mid a_p$. We then need only put $\frac{2a_p}{3}$ copies of S_3 at each vertex in A_1, grouping the ends in threes. If $G = 2K_{1,1,a_p}$, and $3 \mid m$ we must have $a_p \equiv 1 \mod 3$ and

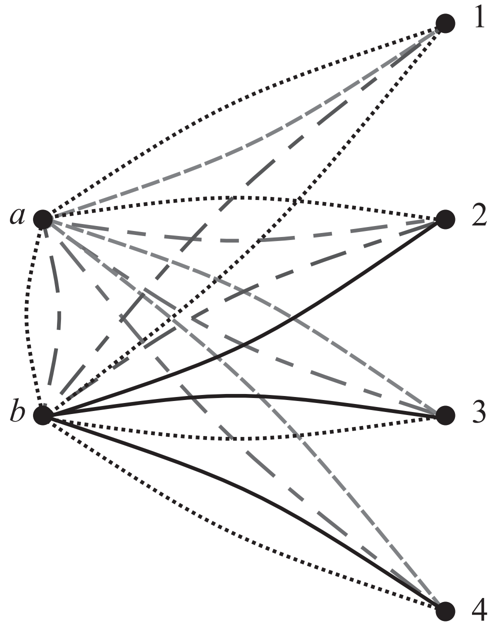

Figure 3.2: An S_3-decomposition of $2K_{1,1,4}$.

so $a_p \geq 4$. We may, then, construct an S_3-decomposition on G as follows: we will label the vertices as $A_1 = \{a\}$, $A_2 = \{b\}$, and $A_p = \{v_1, v_2, \ldots, v_{a_p}\}$. Then we may center three stars at a with ends $\{b, 1, 2\}, \{1, 3, 4\}$, and $\{2, 3, 4\}$. Then center three stars at b with ends $\{b, 1, 2\}, \{1, 3, 4\}$, and $\{2, 3, 4\}$. All edges between A_1 and A_2 are covered, as are the edges between $A_1 \cup A_2$ and four vertices of A_p. If $a_p \geq 7$, the number of vertices in A_p whose edges incident to $A_1 \cup A_2$ have not yet been covered is a multiple of 3 and may be covered in a straightforward manner by grouping ends in threes. Figure 3.2 shows this S_3-decomposition on $2K_{1,1,4}$.

□

We have now have necessary and sufficient conditions for an S_3-decomposition on $2K_{a_1,\ldots,a_p}$. With $\lambda = 2$, many of the inequalities were as tight as they could be; in particular, in the cases where $\min\{q_i, 2\} = q_i$ there were not many possibilities for values of q_i. It was also helpful in many cases that we could assume $n - s \leq 2 = \lambda$. This will be the main difficulty in the next chapter where we examine S_3-decompositions for $\lambda K_{a_1,\ldots,a_p}$ where $\lambda \geq 3$.

Chapter 4

Partial Results for $k = 3,\ \lambda > 2$

Here we examine the problem of S_3-decompositions of λ-fold complete multipartite graphs where $\lambda \geq 3$. We have similar necessary conditions here as we did in the previous case where $\lambda = 2$, along with one small exception to give another necessary condition.

Conjecture 4.1. *Let $G = \lambda K_{a_1,\ldots a_p}$. G has an S_3-decomposition if and only if:*

1. $3 \,|\, m$
2. a. $3(n - a_p) \leq m$ *if* $2a_p \geq n$

   b. $n + \frac{\varepsilon}{\lambda}(n - 2a_p) \leq \frac{2m}{3}$ *where* $\varepsilon = \lambda \mod 2$ *if* $2a_p \leq n$
3. *If* $G = \lambda K_{1,1,a_p}$ *and* λ *is odd, then* $3 \,|\, \lambda$.

The necessity of Conditions 1 and 2 were proved in Theorem 2.2. The proof for the necessity of Condition 3 will be similar to Tazawa's proof of necessity for his Condition 3 in Theorem 1.1, though of course we will have to consider higher values of λ.

Let $G = \lambda K_{1,1,a_p}$ and suppose that G has an S_3-decomposition. Here $C_p = 0$; we cannot center any stars in A_p. The copies of S_3 centered in either A_1 or A_2 must cover all λa_p edges between then and A_p. We then have, for all $i < p$ that

$$\left\lceil \frac{\lambda a_p}{3} \right\rceil \leq q_i \leq \left\lfloor \frac{\lambda(a_p + 1)}{3} \right\rfloor \tag{4.1}$$

We will assume here that λ is odd and either $\lambda \equiv 1 \mod 3$ or $\lambda \equiv 2 \mod 3$. Then by Condition 1 we must have $3 \,|\, m = 2a_p + 1$; thus $a_p \equiv 1 \mod 3$. If $\lambda \equiv 1 \mod 3$ then 4.1 above becomes

$$\left\lceil \frac{\lambda a_p}{3} \right\rceil = \frac{\lambda a_p}{3} + \frac{2}{3} \leq q_i \leq \left\lfloor \frac{\lambda(a_p+1)}{3} \right\rfloor = \frac{\lambda(a_p+1)}{3} - \frac{1}{3} = \frac{\lambda a_p}{3}$$

And so we have a contradiction. The case where $\lambda \equiv 2 \mod 3$ is similar. Thus, $3 \,|\, \lambda$.

Unfortunately we are not, at this time, able to prove that these conditions are always sufficient; that is, for all of the possible subsets $S \subseteq V(G)$ that either $f(S) \geq 0$ or that assuming that set minimizes f, that there is a contradiction. We present the cases that we have finished and comment on the ones still open.

Case 1 ($q_i < \lambda$ for all $i \in I$): We will first examine the case where the central function is constant.

Case 1.1 ($q_i = q_j$ for all $i, j \in I$) We may assume that $V \neq \varnothing \neq W$, or else we would have $S = G$ or $S = \varnothing$. Let $q = \frac{\lambda m}{3n}$ be the value of the central function on all vertices. Here $f(S) = e(S) - qs[3 - (n-s)]$; if $n - s \geq 3$ then $f(S) \geq 0$ so we will assume that $n - s < 3$. By 2.4 and 2.5 we have

$$\frac{\lambda - q}{q} a_w > a_v \tag{4.2}$$

for all $w \in W$ and $v \in V$. Since the central function is constant on all vertices we must have $q \geq \frac{\lambda}{2}$ and so $a_w > a_v$. Also, since the greedy central function respects the capacities of each part and $q > 0$, S must consist of at least two parts. First we will examine the case when $n - s = 2$.

Case 1.1.1 ($n-s=2$) Here $f(S)=e(S)-qs=\lambda m_S-qs$. Since $q<\lambda$ here by assumption and $a_w \geq 2$ for all $w \in W$, by Lemma 2.2 $m_S \geq s \geq \frac{q}{\lambda}s$ and thus $f(S)\geq 0$.

Case 1.1.2 ($V=\{1\}$ and $a_1=1$) Here we have $f(S)=e(S)-2qs$. Let a_{w_0} be the least size of a partite set in S. If $a_{w_0}\geq 4$, then again by Lemma 2.2 $m_S \geq 2s \geq \frac{2q}{\lambda}s$ and thus $f(S)\geq 0$. We must then consider the cases when a_{w_0} is equal to either 2 or 3.

Case 1.1.2.1 ($a_{w_0}=2$) By 4.2 above, $2>\frac{q}{\lambda-q}$ so $q=\frac{\lambda m}{3n}<\frac{2\lambda}{3}$. Now, if S consists of three or more parts, by Lemma 2.2 $m_S\geq\frac{2q}{\lambda}s$ and so $f(S)\geq 0$; we will assume that S is bipartite and so $G=\lambda K_{1,2,n-3}$. However, then $m=3n-7=\frac{3}{\lambda}qn$, which implies that $3n\left(\frac{\lambda-q}{q}\right)=7\geq\frac{3}{2}n$, which is only true when $n\leq 4$, a contradiction.

Case 1.1.2.2 ($a_{w_0}=3$) As in the previous case, we will assume that S is bipartite and further assume that $a_p\leq 5$ as $m_S\geq 3(s-3)\geq 2s$ when $s\geq 9$. By 4.2, $3>\frac{q}{\lambda-q}$ so $q=\frac{\lambda m}{3n}<\frac{3\lambda}{4}$. But if S is bipartite then $G=\lambda K_{1,3,n-4}$ so $m=4n-13$ and $q=\frac{\lambda(4n-13)}{3n}<\frac{3\lambda}{4}$ implies that $n<6$. With S consisting of at least two parts of size at least three, we have a contradiction.

Case 1.2 ($q_i\leq\lambda$ for all $i\in I$, $q_i=q_j+1$ for some $i,j\in I$) These cases are the ones that prevent this conjecture from being a theorem at this time. In the $\lambda=2$ case, the necessary conditions in the related cases there implied that $1\leq q_i\leq 2$ for all i. However, increasing lambda stretches out this set of inequalities, and we have $\frac{\lambda}{2}$ possible central function values to consider. In particular, in the case where $\lambda=2$ the fact that the greater of the two values in the central function was equal to λ was particularly helpful. In these cases, however, the inequalities in section 2.3.1 alone are not enough to draw any conclusions toward proving that either $f(S)\geq 0$ or that a given set cannot minimize f.

Case 2 ($q_i\geq\lambda$ for all $i\in I$) As before, we will first examine the case when the central function is constant.

Case 2.1 ($q_i = q_j$ for all $i, j \in I$) As in other cases where the central function is constant, we must have $\overline{U} = \varnothing$, and if $v = \varnothing$ or $W = \varnothing$ then $S = V(G)$ or $S = \varnothing$ respectively. Thus we may assume $W \neq \varnothing \neq V$. The inequalities 2.4 and 2.5 give $q_w - q_v \geq \frac{\lambda a_v}{3} + \frac{1}{3} \geq 0$, a contradiction.

Case 2.2 ($q_i > q_j$ for some $i, j \in I$)

Case 2.2.1 ($W \neq \varnothing \neq V$) First we will examine the case when $\overline{U} = \varnothing$.

Case 2.2.1.1 ($\overline{U} = \varnothing$ or $\overline{U} = Z$) In this case we will need look at f a little bit differently; we will put f in terms of $\overline{S}$. Here we have

$$\begin{aligned} f(S) &= e(S) - 3c(S) + \sum_{\substack{x \in S \\ y \notin S}} \mu(xy) \\ &= e(S) - 3c(S) + e(S\overline{S}) \\ &= e(G) - e(\overline{S}) - 3c(S) \\ &= 3c(G) - e(\overline{S}) - 3c(S) \\ &= 3c(\overline{S}) - e(\overline{S}) \end{aligned}$$

If $\overline{S}$ consists of only one part, then $e(\overline{S}) = 0$ and $f(S) \geq 0$. Let us then assume that $\overline{S}$ consists of two or more parts. Inequalities 2.4 and 2.5 give us

$$q_w - q_v \geq \frac{\lambda a_v}{3} + \frac{1}{3} > 0$$

Thus, $q_w > q_v$ and as c is decreasing, $a_v \geq a_w$ for all $v \in V$ and $w \in W$; but then $p \in V$. With our assumption that $a_p \geq 2$ and $\lambda \geq 3$, we then have $q_w - q_p \geq 2$. The greedy central function will only give two parts with such a difference in values if either every part in V is at capacity.

We will now show that $3c(\overline{S}) \geq e(\overline{S})$. Let $\overline{S} = \bigcup_{i=\beta+1}^{p} A_i$ and $\overline{s} = |\overline{S}|$. Then

$$\begin{aligned} 3c(\overline{S}) = 3\sum_{i=\beta+1}^{p} \left\lfloor \frac{\lambda(n-a_i)}{3} \right\rfloor a_i &\geq 3\sum_{i=\beta+1}^{p} \left[\frac{\lambda(n-a_i)}{3} - \frac{2}{3} \right] a_i \\ \geq \sum_{i=\beta+1}^{p} [\lambda(n-a_i) - 2a_i] &= \lambda \sum_{i=\beta+1}^{p} (n-a_i)a_i - 2\overline{s} \end{aligned}$$

With $q_\beta - q_{\beta+1} \geq 2$, $A_{\beta+1}$ must be at capacity and $C_\beta > C_{\beta+1}$ so $a_{\beta+1} > a_\beta$; by Lemma 2.2 we have $e(\overline{S}) \geq 2\overline{s}$. Finally,

$$\begin{aligned} \lambda \sum_{i=\beta+1}^{p} (n-a_i)a_i - 2\overline{s} &\geq \lambda \sum_{i=\beta+1}^{p} (n-a_i)a_i - e(\overline{S}) \\ &= 2e(\overline{S}) + e(S\overline{S}) - e(\overline{S}) \\ &= e(S\overline{S}) + e(\overline{S}) \geq e(\overline{S}) \end{aligned}$$

The case where $\overline{U} = Z$ is similar, as $\overline{S}$ consists only of parts whose indices are in V.

Case 2.2.1.2 ($\overline{U} = X$ or $\overline{U} = Y$) From inequalities 2.5 and 2.6 we have $q_w - q_x \geq \frac{\lambda}{3}a_x + \frac{1}{3}$. Since $a_x \geq 2$, $q_w - q_x \geq 2$, but $q_x + 1$ is the maximum value that the central function will take, so we have a contradiction. The case where $\overline{U} = Y$ is similar.

Case 2.2.2 ($W = \varnothing$) In this case, we must have $V = \varnothing = \overline{U}$. Also, we may assume that $\overline{U} \neq X$ as then $S = \varnothing$. Suppose that $\overline{U} = Y$. Here $f(S) = -3(q_y+1)(a_y-b_y)+\lambda(a_y-b_y)(n-a_y)$, so we need $\lambda(n-a_y) \geq 3q_y + 3$ or $q_y \leq \frac{\lambda(n-a_y)}{3} - 1$; this is always the case as $q_y + 1$ is the maximum value the greedy central function will take, and as it respects the capacities of each part, $q_y + 1 \leq \left\lfloor \frac{\lambda(n-a_y)}{3} \right\rfloor$. Thus $f(S) \geq 0$. The case where $\overline{U} = Z$ is similar.

Case 2.2.3 ($V = \varnothing$) Here we will assume $W = \varnothing = \overline{U}$ and $\overline{U} \neq Z$ as that would mean $S = V(G)$. From the inequalities 2.6 and 2.5 we have $q_w - q_x \geq \frac{\lambda}{3}b_x + \frac{1}{3}$ and with our assumption that $\lambda \geq 3$, we have $q_w - q_x \geq 2$. However, by Lemma 2.1 the maximum value that the greedy central function can take is $q_x + 1$, so we have a contradiction. The case where $\overline{U} = Y$ is similar.

Case 3 ($n - a_p < 3$) As in the $\lambda = 2$ case, we will use a different central function here and not the usual greedy one. We have from Condition 2a. that $3(n - a_p) \leq m$, thus guaranteeing that $a_p \geq 3$. If G is bipartite, then this along with Condition 1 is sufficient. If $G = \lambda K_{1,a_p}$ or $\lambda K_{2,a_p}$, we must have $3 \mid a_p$. We then need only put $\frac{\lambda}{3}a_p$ copies of S_3 at each vertex in A_1, grouping ends in threes.

Things get a bit more complicated in the case where $G = \lambda K_{1,1,a_p}$. If λ is even, we may simply use $\frac{\lambda}{2}$ copies of the same S_3-decomposition in Case 4 of Theorem 3.1. We have shown necessity for Condition 3; for sufficiency we will present an S_3-decomposition on $G = 3K_{1,1,a_p}$ which may then be copied as needed.

Now suppose λ is odd and $3 \mid \lambda$. We will separate cases by values of $a_p \mod 3$. In each case we will label the vertices as $A_1 = \{a\}$, $A_2 = \{b\}$, and $A_p = \{1, 2, \ldots, a_p\}$.

Case 3.1 ($a_p \equiv 0 \mod 3$) As $a_p \geq 3$, we will fill A_1 to capacity first; the remainder of the edges will have a straightforward S_3-decomposition. Center 4 copies of S_3 at a with ends $\{b, 1, 2\}, \{b, 1, 3\}, \{b, 2, 3\}$, and $\{1, 2, 3\}$. The number of vertices in A_p whose edges incident to a have not yet been covered is a multiple of 3 and may be covered in a straightforward manner by grouping ends in threes. All vertices from A_1 to $A_2 \cup A_p$ have now been covered. As $3 \mid a_p$ we may easily cover the edges from b to A_p with $3a_p$ copies of S_3.

Case 3.2 ($a_p \equiv 1 \mod 3$) Here we must have that $a_p \geq 4$. As in the previous case, we fill A_1 to capacity first. Center five copies of S_3 at a with ends $\{b,1,2\}, \{b,3,4\}, \{1,2,3\}, \{b,1,4\}$, and $\{2,3,4\}$. Now center four copies of S_3 at b with ends $\{1,2,3\}, \{1,2,4\}, \{1,3,4\}$, and $\{2,3,4\}$. All edges between A_1 and A_2 as well as the edges from four of the vertices of A_p to $A_1 \cup A_2$ have now been covered. If $a_p \geq 7$, the number of vertices in A_p whose edges incident to $A_1 \cup A_2$ have not yet been covered is a multiple of 3 and may be covered in a straightforward manner by grouping ends in threes.

Case 3.3 ($a_p \equiv 2 \mod 3$) Here we must have that $a_p \geq 5$. We will fill A_1 to capacity first; the remainder of the edges will have a straightforward S_3-decomposition. Center at a copies of S_3 with ends $\{1,2,3\}$ and $\{b,4,5\}$; repeat twice more. Now center five copies of S_3 at b with ends $\{1,2,3\}, \{1,4,5\}, \{2,3,4\}, \{1,2,5\}$, and $\{3,4,5\}$. All edges between A_1 and A_2 as well as the edges from five of the vertices of A_p to $A_1 \cup A_2$ have now been covered. If $a_p \geq 8$, the number of vertices in A_p whose edges incident to $A_1 \cup A_2$ have not yet been covered is a multiple of 3 and may be covered in a straightforward manner by grouping ends in threes.

While many of the cases here are similar to those of Theorem 3.1, higher values of λ can be problematic. In the cases where $q_i \leq \lambda$, even with the lower bound on q_i from the necessary conditions, we had $\frac{\lambda}{2}$ different values of the central function to consider. Even in Case 2.3.2 of Theorem 3.1 the greedy central function would not yield an S_3-decomposition. We were then able to come up with a central function that worked, and we suspect the same will need to be done in similar cases for $\lambda \geq 3$.

Chapter 5
Conclusion

We have only begun to scratch the surface of solving the most general problem of finding S_k-decompositions of λ-fold complete multipartite graphs. Using Theorem 2.1 and the fact that all vertices in a part of $\lambda K_{a_1,\ldots,a_p}$ have the same degree, we were able to lay a foundation that we believe will helpful for solving the overall general problem of S_k-decompositions of λ-fold complete multipartite graphs for higher values of k and λ. We then developed a system of inequalities that, given a largest set S that minimizes $f(S)$ gave many necessary conditions on G, often leading to direct contradictions. We summarize here the results we proved.

Proposition 5.1. *We have necessary conditions that are easily checked for finding S_k-decompositions of λ-fold complete multipartite graphs. These conditions are also sufficient for $k = 2$ and all values of λ and for $k = 3$ and $\lambda = 2$.*

In our attempt at the case where $k = 3$ and $\lambda \geq 3$, we ran into difficulty in the cases where $q_i \leq \lambda$ for all i. It is likely, as in the related cases in Theorem 3.1, that we will need to develop another central function to get the desired result. We have begun some preliminary work in finding S_k-decompositions for $2K_{a_1,\ldots,a_p}$ where $k \geq 4$. The main difficulty in this case is, again, when $q_i \leq \lambda = 2$. In most of those cases in the case where $k = 3$, we could assume that $n - s < 3$, which only left a small handful of possible subcases. A similar assumption is not nearly as helpful for larger values of k, especially in the case where $n - a_p < k$. We summarize the open cases remaining.

Proposition 5.2. *For $k = 3$, the problem of finding easily checked necessary and sufficient conditions for S_k-decompositions of λ-fold complete multipartite graphs is still open for graphs where $\lambda \geq 3$, the greedy central function is not constant, and $c(x) \leq \lambda$ for all $x \in V(G)$. It is also open for $\lambda \geq 2$ and $k \geq 4$.*

During our investigation, we developed a small computer program to search for graphs with particular properties. We have included the source code for our program that finds the set(s) that minimize $f(S)$ for graphs with a given number of vertices in the Appendix. Using this program, we are developing some ideas on how to proceed and what types of graphs we may need to try a new central function on.

Bibliography

[1] West, Douglas B. "Introduction to Graph Theory," First Edition. Prentice-Hall (1996).

[2] Tarsi, Michael. "Decomposition of complete multigraphs into stars," Discrete Math. 26 (1979), no. 3, 273 – 278.

[3] Tazawa, Shinsei. "Decomposition of a complete multi-partite graph into isomorphic claws," SIAM J. Alg. Disc. Math. 6 (1985), no. 3, 413 – 417.

[4] Priesler, Miri; Tarsi, Michael. "Multigraph decomposition into stars and into multi-stars," Discrete Math. 296 (2005), no. 2-3, 235 – 244.

[5] Edmonds, Jack. "Paths, trees, and flowers," Canad. J. Math. 17 (1965) 449 – 467.

[6] Hoffman, D.G. "The Real Truth About Star Designs," Discrete Math. 284 (2004) 177 – 180.

Appendix

We include here the source code for a program written in C++ to examine when a k-star design on a graph G is possible. Various adaptations of this program were also used to find graphs with specific properties.

```
#include <iostream.h>
#include <math.h>

// Prototype for function that will use graph to figure things out

int kstars(int a[100], int n, int p, int lambda);

int main(int argc, char* argv[])
{
int n, p, i, x, s, y, lambda;
int a[100];
bool alldone = false;

/*
for (n = 4; n <= 20; n++)
{
p = n-1;
for (i=1; i<=p-1; i++) a[i] = 1;
a[p] = 2;

cout << "What about lambda?";
cin >> lambda;

// Now do what needs to be done on the graphs with n vertices

while (!alldone)
{
// Call function to figure out everything!

kstars(a, n, p, lambda);

// Compute the next graph to be worked on if needed

if (p > 1)
{
s = a[p - 1] + a[p];
x = a[p - 1];
y = floor(s / (x + 1));
for (i = p-1; i <= p-2 + y; i++) a[i] = x + 1;
p = p - 2 + y;
a[p] = a[p] + s - y*(x+1);

}
else alldone = true;
}
alldone = false;
}

kstars(a, n, p, lambda);
}

int kstars (int a[100], int n, int p, int lambda)
{
int i, j, m, e_G, k, t, starsLeft, beta, b_beta, stopPart, f, c_S, e_S,
        minsum, s, min_f, num_with_min, splittype, num_at_cap, g;
int cap[100], q[100], S[100], min_splittype[256];
int sets_with_min[256][256];
bool splitPart, donewithsets, foundone;

// Compute m (number of edges)

m = n*n;
for (i=1; i <= p; i++)
{
m -= a[i]*a[i];
}
m = m/2;
e_G = m * lambda;

// Print the graph and number of edges

cout << "\nG = ";

for (i=1; i<=p; i++)
{
cout << a[i] << " ";
}

// Start looking through the possible values of k for something
// that might work.
// We'll start at k=3 as k<2 is done and/or trivial

    for (k=3; k <= n - a[p]; k++)
{
foundone = false;
```

```
// Check condition 1

if (e_G == 0 || e_G % k != 0)
{
// cout << ": Condition 1 failed for k = " << k << ".\n\n";
continue;
}

// Condition 2?

if (n - a[p] >= k && k*(n - a[p]) > m)
{
// cout << ": Condition 2 failed for k = " << k << ".\n\n";
// continue;
}

// This looks OK so far, so let's compute the capacities of each part

for (i=1; i <= p; i++)
{
if (n-a[i] >= k) cap[i] = floor(lambda*(n-a[i])/k);
else cap[i] = 0;
}

// Now assign our greedy central function to this graph,
// going part by part.We'll first see if we have enough stars
// for the whole part. If so, go ahead and give every vertex
// in this part a star. If there are some left but not
// enough to go around to every vertex, give out what we can
// and then make a note of the part number where we had to stop
// (beta) and where in the part we had to stop (a[beta]-b_beta).

starsLeft = e_G / k;

// Reset the q's to all zeros

for (i=1; i<=p; i++) q[i] = 0;

// stopPart is the last part that isn't at capacity (and therefore
// where we want to stop giving out stars).

stopPart = p;
splitPart = false;
b_beta = 0;

do
{
```

```
for (i=stopPart; cap[i] ==  q[i] && i > 1; i--);

stopPart = i;

for (i=1; i<=stopPart && a[i] <= starsLeft; i++)
{
q[i] += 1;
starsLeft -= a[i];
}

// This checks to see if we didn't have enough stars

if (a[i] > starsLeft && starsLeft > 0 && i <= stopPart)
{
beta = i;
b_beta = a[beta] - starsLeft;
starsLeft = 0;
splitPart = true;
}


} while (starsLeft > 0);

// Now that the greedy central function is set, we need to
// check condition ii). For the graphs we're looking at though,
// it's not too bad. We need only check to see if the values of
// the central function in the last two parts add to lambda.

if (q[p] + q[p-1] < lambda)
{
cout << ": Condition 2 failed for k = " << k << ". Moving on...\n";
continue;
}

// This prints out the central function.

num_at_cap = 0;

for (i=1; i<=p; i++) if (q[i] == cap[i] && cap[i] > 0) num_at_cap++;

cout << "\nOur central function for k = " << k << ":\n";
for (i=1; i<= p; i++) cout << "q[" << i << "] = " << q[i]
            << "  capacity = " << cap[i] << "\n";
if (splitPart)  cout << "beta = " << beta << ", b_beta = "
            << b_beta << ", a[beta] = " << a[beta] << "\n"; */
```

```
// Now we're going to check this to see what subset of parts
// minimizes the inequality in condition iii). To do this, we're
// going to have to look at all possible subsets.

// To start off with, we need to set up an array, S, that will
// keep track of what parts are in the set we're looking at (1)
// and aren't (0). First we need to reset
// S to all zeros.

for (i=1; i<=100; i++)
for (j=1; j<=p; j++) sets_with_min[i][j] = 0;
min_f = 0;
num_with_min = 0;
for (i=1; i<=100; i++) min_splittype[i] = 0;

if (!splitPart)
{
for (i=1; i<=p; i++) S[i] = 0;
donewithsets = false;
while (!donewithsets)
{
donewithsets = true;
f = 0;
c_S = 0;
e_S = 0;
minsum = 0;
s = 0;

for (i=1; i<=p && donewithsets == true; i++) if (S[i] == 0)
                    donewithsets = false;

for (i=1; i<=p; i++)
{
if (S[i] == 1)
{
c_S += (a[i]*q[i]);
s += a[i];
e_S -= pow(a[i],2);
if (q[i] < lambda) minsum += a[i]*q[i];
else minsum += (a[i]*lambda);
}
}

e_S += pow(s,2);
e_S = e_S * lambda / 2;
minsum *= (n - s);
```

```
f = e_S - k*c_S + minsum;

// Now we need to see if this is less than the minimum
// value of f we've reached so far.

if (f < min_f)
{
min_f = f;
num_with_min = 1;
for (j=1; j<=p; j++) sets_with_min[1][j] = S[j];
}
else if (f == min_f)
{
num_with_min++;
for (j=1; j<=p; j++) sets_with_min[num_with_min][j] = S[j];
}

// If this isn't the last subset (namely, everything)
// then compute the next set

if (!donewithsets)
{
t=p;
while (S[t] != 0) t--;
S[t] = 1;
for (i=t+1;i<=p;i++) S[i] = 0;
}

}

}
else
{
// Now things get a little trickier. We've got a split part,
// so we're going to have to look at three possibilities:
// the split part being in X, Y, or Z.
// The int splittype variable keeps track of what
// case we're looking at:

// 0 = X
// 1 = Y
// 2 = Y'
// 3 = Z

for (splittype=0; splittype <=2; splittype++)
{
for (i=1; i<=p; i++) S[i] = 0;
```

```
donewithsets = false;
while (!donewithsets)
{
donewithsets = true;
f = 0;
c_S = 0;
e_S = 0;
minsum = 0;
s = 0;

for (i=1; i<= p-1 && donewithsets == true; i++)
                        if (S[i] == 0) donewithsets = false;

for (i=1; i<beta; i++)
{
if (S[i] == 1)
{
c_S += (a[i]*q[i]);
s += a[i];
e_S -= pow(a[i],2);
minsum +=  (((q[i]) < (lambda)) ? (q[i]) : (lambda)) * a[i];
}
}
for (i=beta; i<=p-1; i++)
{
if (S[i] == 1)
{
c_S += (a[i+1]*q[i+1]);
s += a[i+1];
e_S -= pow(a[i+1],2);
minsum += (((q[i+1]) < (lambda)) ? (q[i+1]) : (lambda)) * a[i+1];
}
}

minsum *= (n - s);

switch (splittype)
{
case 0:
;
break;
case 1:
c_S += (q[beta] + 1)*(a[beta] - b_beta);
s += (a[beta] - b_beta);
e_S -= pow((a[beta] - b_beta),2);
minsum += (((q[beta]+1) < (lambda)) ? (q[beta]+1) :
                            (lambda)) * (a[beta] - b_beta) * (n - s - b_beta);
```

```
break;
case 2:
c_S += (q[beta] * b_beta);
s += b_beta;
minsum += ((((q[beta]) < (lambda)) ? (q[beta]) :
                            (lambda)) * (b_beta) * (n - s - (a[beta] - b_beta)));
break;
case 3:
c_S += a[beta]*q[beta] + (a[beta] - b_beta);
s += a[beta];
e_S -= pow(a[beta],2);
minsum += (((q[beta]<lambda) ?  (n - s)*a[beta]*q[beta]
                            + (a[beta] - b_beta) : (n - s)*(a[beta]*lambda)));
break;
default:
cout << "Error checking for split parts. Exiting.";
return 1;
}

e_S += pow(s,2);
e_S = e_S * lambda / 2;
f = e_S - k*c_S + minsum;

// Now we need to see if this is less than the minimum value of f
// we've reached so far. Also keep track of the min's split type.

if (f < min_f)
{
min_f = f;
num_with_min = 1;
for (j=1; j<=p-1; j++) sets_with_min[1][j] = S[j];
min_splittype[1] = splittype;
}
else if (f == min_f)
{
num_with_min++;
for (j=1; j<=p-1; j++) sets_with_min[num_with_min][j] = S[j];
min_splittype[num_with_min] = splittype;
}

// If this isn't the last subset (namely, everything)
// then compute the next set

if (!donewithsets)
{
t=p-1;
while (S[t] != 0) t--;
```

```
S[t] = 1;
for (i=t+1;i<=k;i++) S[i] = 0;
}
}
}
}

// Print the results out to the screen

 cout << "For k = " << k << ", The minimum value of f is " << min_f << "\n";

for (i=1; i<=num_with_min; i++)
{
cout << "Min set " << i << ": ";
if (splitPart)
{
for (j=1; j<beta; j++) cout << sets_with_min[i][j];
switch (min_splittype[i])
{
case 0:
cout << "X";
break;
case 1:
cout << "Y";
break;
case 2:
cout << "Y'";
break;
case 3:
cout << "Z";
break;
default:
cout << "Error checking for split parts. Exiting.";
return 1;
}
for (j=beta; j<= p-1; j++) cout <<  sets_with_min[i][j];
}
else for (j=1; j<=p; j++) cout << sets_with_min[i][j];
cout << endl;
}
}
return 0;
}
```