EDGE-CONNECTIVITY AUGMENTATION WITH PARTITION CONSTRAINTS

Jørgen Bang-Jensen* Harold N. Gabow † Tibor Jordán ‡
Zoltán Szigeti§

July 10, 1997

Abstract

In the well-solved edge-connectivity augmentation problem we must find a minimum cardinality set F of edges to add to a given undirected graph to make it k-edge-connected. This paper solves the generalization where every edge of F must go between two different sets of a given partition of the vertex set. A special case of this partition-constrained problem, previously unsolved, is increasing the edge-connectivity of a bipartite graph to k while preserving bipartiteness. Based on this special case we present an application of our results in statics. Our solution to the general partition-constrained problem gives a min-max formula for $|F|$ which includes as a special case the original min-max formula of Cai and Sun [3] for the problem without partition constraints.

When k is even the min-max formula for the partition-constrained problem is a natural generalization of [3]. However this generalization fails when k is odd. We show that at most one more edge is needed when k is odd and we characterize the graphs that require such an extra edge.

We give a strongly polynomial algorithm that solves our problem in time $O(n(m + n \log n) \log n)$. Here n and m denote the number of vertices and distinct edges of the graph respectively. This bound is identical to the best-known time bound for the problem without partition constraints. Our algorithm is based on the splitting off technique of Lovász, like several known efficient algorithms for the unconstrained problem. However unlike previous splitting algorithms, when k is odd our algorithm must handle "obstacles" that prevent

*Department of Mathematics and Computer Science, Odense University, Campusvej 55, DK-5230 Odense, Denmark. e-mail: jbj@imada.ou.dk

†Department of Computer Science, University of Colorado at Boulder, Boulder, Colorado 80309-0430, USA. e-mail: hal@cs.colorado.edu

‡Department of Mathematics and Computer Science, Odense University, Campusvej 55, DK-5230 Odense, Denmark. Supported in part by the Danish Natural Science Research Council Grant Number 28808. e-mail: tibor@imada.ou.dk

§Equipe Combinatoire, Universite Paris VI, 4, place Jussieu, 75252 Paris, France. e-mail: szigeti@ec.ibp.fr

all edges from being split off. Our algorithm is of interest even when specialized to the unconstrained problem, because it produces an asymptotically optimum number of distinct splits.

1 Introduction

In the *edge-connectivity augmentation problem* we are given an undirected graph $G = (V, E)$ and a positive integer k; the goal is to find a smallest set F of edges on the vertex-set V for which $G' = (V, E \cup F)$ is k-edge-connected. Note that E as well as F may contain parallel edges.

This optimization problem has been extensively investigated, in part due to possible practical applications in the design of reliable networks (see [8], [21]). The first polynomial-time algorithm solving this problem is due to Watanabe and Nakamura [22]. Other approaches were later developed which led to more efficient algorithms. Cai and Sun [3] gave a min-max characterization for the corresponding optimum value, using the splitting off method. "Splitting off" a pair su, sv of edges means replacing su and sv by a new edge uv. Using this method Frank [6] solved several extensions of the problem. For example he showed it is tractable even if local connectivity demands or vertex costs are given. A different approach due to Naor *et al.* [19] resulted in a faster algorithm. The currently fastest algorithm was developed by Nagamochi *et al.* [17], [18]. Their algorithm, also based on the splitting off method, runs in time $O(n(m + n \log n) \log n)$. (The parameters n and m denote the number of vertices and distinct edges of the graph respectively.) For further related results and the directed- and vertex-connectivity versions of the problem see the survey paper [7].

This paper solves a generalization of the above problem which we call *edge-connectivity augmentation with partition constraints*. Here a partition $\mathcal{P} = \{P_1, ..., P_r\}$ ($r \geq 2$) of V is also given and the additional requirement is that each edge of the augmenting set F must connect different classes of $\mathcal{P}$. Taking $r = |V|$ gives the original (or in other words *unconstrained*) edge-connectivity augmentation problem as a special case. Another special case is the version where G is a bipartite graph with bipartition $V = (A, B)$ and $\mathcal{P} = \{A, B\}$. In this problem, which was open so far and was one of our main motivations, the goal is to increase the edge-connectivity of a bipartite graph while preserving bipartiteness.

The first part of our solution to the augmentation problem with partition constraints is a min-max formula for the optimum value (i.e., the minimum size of F). When k is even we show the optimum value equals the "deficiency" of a certain subpartition of V. This subpartition is either the subpartition of Cai and Sun for the unconstrained problem, or a subpartition based on the partition constraints. When k is odd these subpartitions have deficiency equal to the optimum value for "most" graphs but they fail for an infinite number of graphs. The following simple example turns out to be central.

Consider the problem of augmenting a four-cycle to achieve 3-edge-connectivity. It is easy to see that the unique optimum augmentation adds two new edges, changing the graph into K_4. Now consider augmenting a four-cycle to make it 3-edge-connected and also preserve bipartiteness. It is easy to see that for this problem three new edges is optimum. But we will see that the subpartitions have deficiency only two.

Our min-max theorem shows that when k is odd any graph requires at most one more edge than the lower bounds from subpartitions. Furthermore a graph requires an extra edge precisely when it possesses a structure generalizing the above four-cycle example, or a structure generalizing the six-cycle. These structures, which we call C_4- and C_6- configurations, can exist for any odd value $k \geq 3$. A C_6-configuration is more specialized, since it only exists in graphs requiring exactly three edges to achieve k-edge-connectivity.

The second part of our solution to the partition-constrained problem is a strongly-polynomial algorithm. It finds an optimum solution in the same time as the algorithm of Nagamochi *et al.* for the unconstrained problem, $O(n(m + n \log n) \log n)$. When specialized to the unconstrained problem our algorithm is a modified version of the splitting-off algorithm of [18]. This specialization is also of interest because, like our general algorithm, it produces an asymptotically optimum number of distinct splits. This improves [17], [18] by a logarithmic factor. It gives a similar savings in space.

The splitting off method used in algorithms for edge-connectivity-augmentation and a number of other problems is based on the well-known splitting off theorem of Lovász [14]. The partition-constrained problem requires a more powerful result: When k is odd we must avoid certain splits that would mistakenly force an extra edge to be added. We give a new splitting off theorem that characterizes when this can be done.

This paper also investigates an application of the problem of augmenting the connectivity of a bipartite graph (preserving bipartiteness) in the field of statics. It also shows that a variant of our problem, when the edges of the augmenting set F must lie *within* classes of a given partition, is NP-hard.

We close this section by mentioning other extensions of the edge-connectivity augmentation problem that have been investigated. Finding a k-edge-connected augmentation that has minimum cost is NP-complete (even when $k = 2$ and there are only two distinct edge costs), see [5]. Optimally augmenting a bipartite graph to achieve 2-vertex-connectivity while preserving bipartiteness was solved in [10]. Optimally augmenting to achieve k-edge-connectivity preserving simplicity of the graph is NP-hard, but polynomially solvable for fixed k, see [1] and [11]. Preserving planarity was investigated in some vertex-connectivity problems, see [13]. Other types of constraints were also studied. In [4] an optimal augmenting set F is to be found which can be extended to an optimal augmentation with respect to an arbitrary higher target.

Section 2 contains definitions and some basic results. Section 3 proves the new splitting off theorem. The results on the augmentation problem are in Sections 4 – 5. Our efficient algorithm is in Section 6. The application to statics is in Section 7 and the NP-hardness result is in Section 8.

2 Terminology and some basic results

Let $G = (V, E)$ be an arbitrary undirected multigraph. A *subpartition* of V is a collection of pairwise disjoint subsets of V. The subgraph of G induced by a subset X of vertices is denoted $G[X]$. A set consisting of a single vertex v is simply denoted by v. An edge joining vertices x and y is denoted xy. Sometimes xy will refer to an arbitrary copy of the parallel edges between x and y but this will not cause any confusion. Adding or deleting an edge e from a graph G is often denoted by $G + e$ or $G - e$, respectively.

For $X, Y \subseteq V$, $d(X, Y)$ denotes the number of edges with one endvertex in $X - Y$ and the other in $Y - X$. We define the degree of a subset X as $d(X) = d(X, V - X)$. For example $d(v)$ denotes the degree of vertex v. The degree-function of a graph G' is denoted d'. A graph

$G = (V, E)$ is k*-edge-connected* if

$$d(X) \geq k \quad \text{for all } \emptyset \neq X \subset V. \tag{1}$$

The degree function satisfies the following two well-known equalities.

Proposition 2.1 *Let $H = (V, E)$ be a graph. For arbitrary subsets $X, Y \subseteq V$,*

$$d(X) + d(Y) = d(X \cap Y) + d(X \cup Y) + 2d(X, Y), \tag{2}$$

$$d(X) + d(Y) = d(X - Y) + d(Y - X) + 2d(X \cap Y, V - (X \cup Y)). \tag{3}$$

The operation *splitting off* a pair of edges sv, st ($v \neq t$) at a vertex s means that we replace these two edges by a new edge vt. (In the presence of parallel edges one splitting operation replaces only one copy of each of sv, vt.) The notation $G_{v,t}$ denotes the graph obtained after splitting off the edges sv, st in G (the vertex s will always be clear from the context). A *complete splitting* at a vertex s (with even degree) is a sequence of $d(s)/2$ splittings of pairs of edges incident to s.

In the rest of this section let s be a specified vertex of a graph $G = (V + s, E)$ with degree function d such that $d(s)$ is even and (1) holds. We shall consider such graphs throughout the paper. Saying (1) holds in such a graph G means it holds for all $\emptyset \neq X \subset V$. (So (1) can hold even when $d(s) < k$.) A set $\emptyset \neq X \subset V$ is called *dangerous* if $d(X) \leq k + 1$ and *critical* if $d(X) = k$. Two sets $X, Y \subseteq V$ are *crossing* if $X - Y$, $Y - X$, $X \cap Y$ and $V - (X \cup Y)$ are all nonempty. Again these definitions refer only to subsets of V.

Edges sv, st form an *admissible pair* in G if $G_{v,t}$ still satisfies (1). It is easy to see that sv, st is not admissible if and only if some dangerous set contains both t and v.

Throughout the paper we use the following basic lemmas.

Lemma 2.2 *If $k \geq 3$ and $d(X) \leq k + 2$ for some $X \subset V$ then $G[X]$ is connected.*

Proof: Let $X = Y \cup Z$ be any partition of X into two nonempty sets. Since G satisfies (1) we get $k + 2 \geq d(X) = d(Y) + d(Z) - 2d(Y, Z) \geq k + k - 2d(Y, Z)$. Thus since $k \geq 3$, we must have that $d(Y, Z) \geq 1$ and hence the lemma follows. □

Lemma 2.3 *If k is odd then no three pairwise disjoint critical sets X, Y, Z have $N(X) \subseteq Y \cup Z$.*

Proof: Assume on the contrary that there exist such sets. Since $k = d(X) = d(X, Y) + d(X, Z)$, w.l.o.g. we may assume $d(X, Y) > k/2$. It follows that $d(X \cup Y) = d(X) + d(Y) - 2d(X, Y) < k + k - 2k/2 = k$, in contradiction to (1). □

Lemma 2.4 *If k is odd then the union of two disjoint critical sets is not critical.*

Proof: If X and Y are disjoint critical sets then $d(X \cup Y) = d(X) + d(Y) - 2d(X,Y) = 2k - 2d(X,Y)$. The right-hand side does not equal k because k is odd. $\square$

Lemma 2.5 *If k is odd then no two critical sets are crossing.*

Proof: Suppose that X and Y are two crossing sets in V of degree k. Then $d(X \cap Y) = k$ by (2) and $d(X - Y) = k$ by (3). Thus $X - Y$ and $X \cap Y$ contradict Lemma 2.4. $\square$

Lemma 2.6 *A maximal (w.r.t. to inclusion) dangerous set does not cross any critical set.*

Proof: Let X be a maximal dangerous set and suppose it crosses a critical set Y. The maximality of X shows $d(X \cup Y) \geq k+2$. Thus $k+1+k \geq d(X)+d(Y) \geq d(X \cap Y)+d(X \cup Y) \geq k+k+2$, a contradiction. $\square$

Lemma 2.7 *Let k be odd and let $X \subset V$ be a set of degree $k+2$ containing two neighbors of s, x and y, such that $d(Z) \geq k+2$ whenever $x, y \in Z \subset V$. Then X does not cross any critical set.*

Proof: The hypothesis implies sx, sy is an admissible pair. Applying Lemma 2.5 to $G_{x,y}$ gives the desired conclusion. $\square$

Lemma 2.8 *If X is dangerous then $d(s, V - X) \geq d(s, X)$.*

Proof: By (1) we see that $k \leq d(V - X) = d(X) - d(s, X) + d(s, V - X)$. Since $d(X) \leq k+1$, this implies $d(s, V - X) \geq d(s, X) - 1$ and since $d(s)$ is even we cannot have equality. $\square$

Lemma 2.9 *If k is even then two maximal dangerous sets X, Y which are crossing have $d(s, X \cap Y) = 0$.*

Proof: Suppose X and Y are maximal dangerous and crossing such that $d(s, X \cap Y) \geq 1$. By Proposition 2.1 and (1), we get that $d(X) = k+1$, $d(X - Y) = k$ and $d(X \cap Y) = k$. It is easy to see that $d(X)$ is congruent to $d(X - Y) + d(X \cap Y)$ modulo 2, but this contradicts the fact that $d(X)$ is odd and each of $d(X - Y), d(X \cap Y)$ is even. $\square$

Lemma 2.10 *Any two crossing sets $X, Y \subset V$ satisfy $d(X) + d(Y) \geq 2k + 2d(s, X \cup Y) - d(s)$.*

Proof: Observe that $d(X \cup Y) = d(V - (X \cup Y)) - d(s, V - (X \cup Y)) + d(s, X \cup Y)$ and $d(s) = d(s, X \cup Y) + d(s, V - (X \cup Y))$. Combining these and using (1) and (2) gives the lemma. $\square$

Lemma 2.11 *Let X be a critical set that is minimal w.r.t. inclusion. Then for every edge sx, $x \in X$ and every $u \in X$, $G - sx + su$ satisfies (1).*

Proof: Suppose that for some edge $e = sx$, $x \in X$ and some $u \in X$ the graph $G - e + f$, where $f = su$, does not satisfy (1). Then there must be some other critical set Y in G such that the edge sx enters Y while su does not. By the minimality of X we get that $Y - X \neq \emptyset$. Now applying (3) we get that $k + k = d(X) + d(Y) = d(X - Y) + d(Y - X) + 2d(s, X \cap Y) \geq k + k + 2$, a contradiction. □

Throughout the rest of this paper we always assume that $k \geq 2$ since the solutions to our problems are trivial when $k \leq 1$.

The following result of Lovász [14] – Theorem 2.12(a) below – turned out to be a useful tool in augmentation problems, see [6]. Here we formulate an easy extension, Theorem 2.12(b), which plays an important role in some of our arguments. The proof follows from the proof of part (a) given by A. Frank in [6, pages 35-36]. We assume that the reader is familiar with his proof and give only the necessary additional details below.

Theorem 2.12 *Suppose that (1) holds in $G = (V + s, E)$, $k \geq 2$ and $d(s) > 0$ is even. Then (a) [14] for every edge st there exists an edge su such that the pair st, su is admissible.*

(b) for every edge st the number of edges su ($t \neq u$) for which the pair st, su is admissible is at least $d(s)/2$ when k is even and at least $d(s)/2 - 1$ when k is odd.

Proof of (b):

In the proof of (a) in [6, pages 35-36] it is shown that for any given edge st all the neighbours u of s for which the pair st, su is not admissible are contained in the union of at most two maximal dangerous sets, both containing t; furthermore when there are two such sets they are crossing. Hence if k is even, then by Lemma 2.9, either st is admissible with every neighbour of s, or there is a unique maximal dangerous set X which contains t and all neighbours u of s for which st, su is not admissible for splitting. Thus st is admissible with each edge sv where $v \notin X$ and Lemma 2.8 shows there are at least $d(s)/2$ such edges.

Suppose next that k is odd. If t is not contained in any maximal dangerous set or there is a unique maximal dangerous set containing t, then we are done as above. So assume that there are two maximal dangerous sets X, Y such that both contain t. Since X and Y are crossing (as remarked above) we get from (2) that $d(X \cup Y) = k + 2$. Now look at $Z = V - (X \cup Y)$. Since G satisfies (1) we get $k \leq d(Z) = d(X \cup Y) - d(s, X \cup Y) + d(s, Z)$. Since $d(X \cup Y) = k + 2$ this gives that $d(s, Z) \geq d(s, X \cup Y) - 2$ implying that at least $d(s)/2 - 1$ edges incident to s are admissible for splitting with st. □

Let $OPT^k(G)$ ($OPT^k_{\mathcal{P}}(G)$) denote the size of a solution to the edge-connectivity augmentation problem (with partition constraints). (Hence $OPT^k(G)$ equals the minimum size of a new set of edges whose addition to G results in a k-edge-connected graph. $OPT^k_{\mathcal{P}}(G)$ is defined similarly with the additional stipulation that no new edge lies inside some $P_i \in \mathcal{P}$, where $\mathcal{P} = \{P_1, \ldots, P_r\}$ is a given partition of V.)

Our algorithms are based Frank's algorithm to calculate $OPT^k(G)$ and solve the edge-connectivity augmentation problem. Frank's algorithm uses the splitting operation and Theorem 2.12(a) as the main tool. We now summarize Frank's algorithm [6].

Frank's algorithm

(PHASE 1) Starting with the given graph $G = (V, E)$ and integer k, add a new vertex s to V and a set F of new edges between s and some vertices of V such that

$$d'(X) \geq k \text{ for all } \emptyset \neq X \subset V \text{ in } G' = (V + s, E \cup F), \tag{4}$$

$$F \text{ is minimal (w.r.t. inclusion) subject to } (4). \tag{5}$$

Remark: It is clear that such an F exists. It was shown in [6] that

$$\text{there exists a subpartition } \mathcal{F} = \{X_1, ..., X_t\} \text{ of } V \text{ such that } |F| = \sum_1^t (k - d(X_i)) \tag{6}$$

where as usual d denotes the degree function in G.

(PHASE 2) If $d'(s)$ is odd in G', add a new edge sv for some arbitrary $v \in V$.

(PHASE 3) Split off admissible pairs of edges incident to s in arbitrary order, maintaining (1). When s becomes isolated, delete s.

In the third phase every edge can be split off by Theorem 2.12 (a). The resulting graph is an optimal k-edge-connected augmentation of G since after the second phase $OPT^k(G) \geq |F|/2$ by (6).

We now focus on edge-connectivity augmentation with partition constraints. An admissible pair sx, sy is called *allowed* if x and y belong to different classes of $\mathcal{P}$. A *complete allowed splitting* at a vertex s is a sequence of $d(s)/2$ splittings at s such that the ith splitting is allowed when performed in the graph obtained after performing the first $i - 1$ splittings. The following fact was also observed in [18] (in a different context).

Lemma 2.13 *Let k be even and let $G = (V + s, E)$ be a graph satisfying (1) with $d(s)$ even. There exists a complete allowed splitting at vertex s if and only if $d(s, P_i) \leq d(s)/2$ for all $1 \leq i \leq r$.*

Proof: The condition $d(s, P_i) \leq d(s)/2$ is necessary for a complete allowed splitting to exist, since a pair sx, sx is not allowed.

To prove sufficiency we argue by induction on the degree of s. Choose an index j, $1 \leq j \leq r$, so that $d(s, P_j)$ is maximum. Theorem 2.12(b) implies there is an allowed pair sx, sy for any neighbour x of s belonging to P_j. Here we use the hypothesis $d(s, P_j) \leq d(s)/2$. Next we show that for $G' = G_{x,y}$, $d'(s, P_i) \leq d'(s)/2$ for all i. This is clear for $i = j$. If $i \neq j$ then $d'(s, P_i) \leq d'(s, P_j) + 1$. If equality holds then some index $\ell \neq i, j$ has $d'(s, P_\ell) \geq 1$ since $d'(s)$ is even. This implies $d'(s, P_i) \leq d'(s)/2$. □

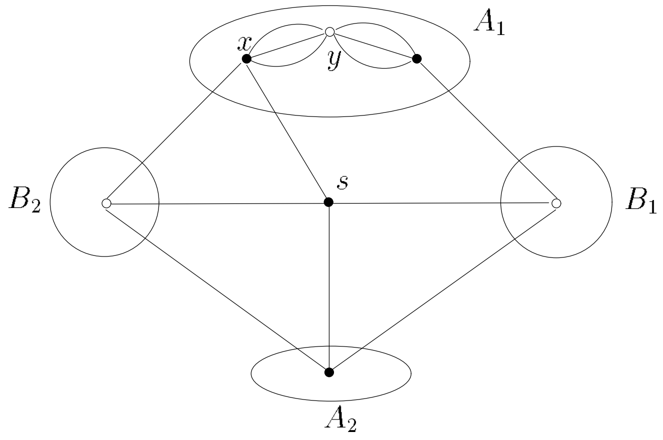

Figure 1: A graph $G = (V + s, E)$ with a partition $\mathcal{P} = \{P_1, P_2\}$ of V, where the black (white) circles indicate vertices in P_1 (P_2). The sets A_1, A_2, B_1, B_2 form a C_4-obstacle in G with respect to $k = 3$.

3 Splittings satisfying partition constraints when k is odd

Let $k \geq 3$ be an odd number, let $G = (V + s, E)$ be a graph such that (1) holds for G and $d(s)$ is even and let $\mathcal{P} = \{P_1, P_2, \ldots, P_r\}$, $2 \leq r \leq |V|$ be a prescribed partition of V.

We begin by defining two types of "obstacles" that, we will show, preclude the existence of a complete allowed splitting. Let S denote the set of neighbours of s and $S_i := S \cap P_i$. Let $d_i := d(s, P_i)$.

Definition 3.1 *Let $A_1 \cup A_2 \cup B_1 \cup B_2$ be a partition of V with the following properties in G for some index i, $1 \leq i \leq r$:*

(i) $d(X) = k$ *for* $X = A_1, A_2, B_1, B_2$;

(ii) $d(X, Y) = 0$ *for* $(X, Y) = (A_1, A_2), (B_1, B_2)$;

(iii) $S \cap X = S_i$ *for* $X = A_1 \cup A_2$ *or* $X = B_1 \cup B_2$;

(iv) $d_i = d(s)/2$.

Such a partition is called a C_4-obstacle *in G. The two pairs (A_1, A_2) and (B_1, B_2) listed in (ii) are called* nonconsecutive *while the other four pairs are called* consecutive. *(See Figure 1.)*

Note that conditions (i) – (iv) can be satisfied when k is even but a C_4-obstacle requires that k be odd. An important property of a C_4-obstacle is that s has a neighbour in each set A_1, A_2, B_1, B_2, by Lemma 2.3.

The reader will have noticed the close relation between a C_4-obstacle and the example of augmenting a four-cycle given in the Introduction. The C_6-obstacle defined below has a similar

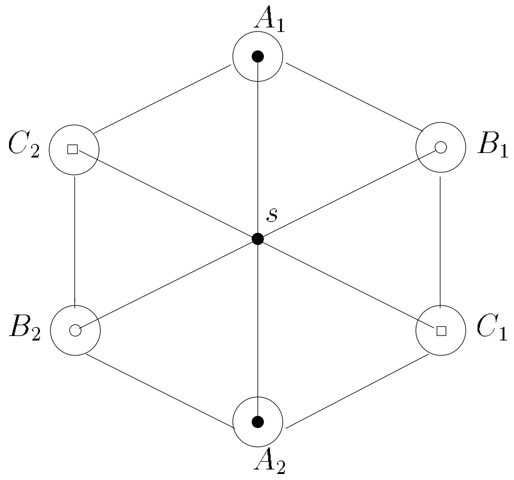

Figure 2: A graph $G = (V + s, E)$ with a partition $\mathcal{P} = \{P_1, P_2, P_3\}$. Each P_i has size 2 and P_1 is indicated by black circles, P_2 by white circles and P_3 by boxes. The sets $A_1, A_2, B_1, B_2, C_1, C_2$ form a C_6-obstacle in G with respect to $k = 3$.

motivation which we now give. Suppose we wish to augment a six-cycle to achieve 3-edge-connectivity. This can be done by adding three new edges to change the graph into $K_{3,3}$. It is easy to see that three new edges is optimum, and any optimum augmentation adds at least one new edge joining diametrically opposite vertices. Now consider augmenting a six-cycle to make it 3-edge-connected with the partition constraint imposed when $\mathcal{P}$ consists of the three sets of diametrically opposite vertices. It is easy to see that for this problem four new edges is optimum. However we will see in Section 5 that the subpartitions give a lower bound of only three.

Definition 3.2 *Let $A_1 \cup A_2 \cup B_1 \cup B_2 \cup C_1 \cup C_2$ be a partition of V with the following properties in G for three distinct indices a, b, c, $1 \leq a, b, c \leq r$:*

(i') $d(X) = k$ *for* $X = A_1, A_2, B_1, B_2, C_1, C_2$;

(ii') $d(X, Y) = (k-1)/2$ *for* $(X, Y) = (A_1, B_1), (B_1, C_1), (C_1, A_2), (A_2, B_2), (B_2, C_2), (C_2, A_1)$;

(iii') $d(X, s) = 1$ *for* $X = A_1, A_2, B_1, B_2, C_1, C_2$;

(iv') $S \cap X = S_i$ *for* $(X, S_i) = (A_1 \cup A_2, S_a), (B_1 \cup B_2, S_b), (C_1 \cup C_2, S_c)$.

Such a partition is called a C_6-obstacle *in G. The six pairs listed in (ii') are called* consecutive *while the other nine pairs are called* nonconsecutive. *(See Figure 2.)*

The first two lemmas allow us to split off most edges.

Lemma 3.3 *Suppose $d(s, P_i) \leq d(s)/2$ for all $i = 1, 2, \ldots, r$. If $d(s) \geq 6$ then for any P_i containing a neighbor of s there is an allowed pair sx, sy with $x \in P_i$.*

Proof: If $d_i < d(s)/2$ then Theorem 2.12 implies there is an allowed pair sx, sy for any neighbour x of s belonging to P_i. Hence assume $d_i = d(s)/2$. Let u be an arbitrary neighbour of s in P_i. If there is no allowed pair for the edge su then, as mentioned in the proof of Theorem 2.12, there are two dangerous sets X and Y so that $u \in X \cap Y$ and $X \cup Y$ contains all the neighbours of s not in P_i. Since $d_i = d(s)/2$ and $d(s) \geq 6$, either $d(s, X \cap (V - P_i)) \geq 2$ or $d(s, Y \cap (V - P_i)) \geq 2$. In both cases there is a pair sy, sz of edges incident to s so that $y, z \notin P_i$ and the edges sy, sz enter a common dangerous set. (It is possible that $y = z$.) Hence sy, sz is not admissible. Theorem 2.12 shows sy is part of at least $d(s)/2 - 1$ admissible pairs. This implies for at least one $x \in P_i$ the pair sx, sy is an allowed splitting. □

Lemma 3.4 *Suppose $d(s, P_i) \leq d(s)/2$ for all $i = 1, 2, \ldots, r$. If $d(s) = 4$ then there exists a complete allowed splitting at s unless G contains a C_4-obstacle. Furthermore if this C_4-obstacle is A_1, A_2, B_1, B_2 then in graph $G - s$,*

1. $d(A_1) = d(A_2) = d(B_1) = d(B_2) = k - 1,$
2. $d(A_1 \cup B_1) = d(A_1 \cup B_2) = k - 1.$

Proof: If $\max d_i = 1$ then every admissible pair is allowed and thus we are done by Theorem 2.12. Otherwise the assumption of the lemma shows $\max d_i = 2$. Assume w.l.o.g. that $d_1 = 2$. We may also assume that there are no parallel edges between the edges incident to s (since in that case we would be done again, by applying Theorem 2.12 twice). Let us denote the two neighbours of s in P_1 by x and y and denote the other two neighbours of s by u and v. Assume that neither sx, su nor sx, sv is admissible. (If either pair is admissible we are again done.) Then Lemma 2.8 implies there are two dangerous sets X and Y so that $x \in X \cap Y$, $u \in X - Y$, $v \in Y - X$ and $y \in V - (X \cup Y)$. Now we show that $X \cap Y, V - (X \cup Y), X - Y, Y - X$ is a C_4-obstacle: Let d denote the degree function in G. (2) implies $d(X \cup Y) = k+2$, $d(X \cap Y) = k$ and $d(X, Y) = 0$. (3) implies $d(X - Y) = d(Y - X) = k$ and $d(X \cap Y, V - (X \cup Y)) = 0$. Also $d(V - (X \cup Y)) = d(X \cup Y) - 2 = k$. We have now shown (i) – (ii) of Definition 3.1 and (iii) – (iv) are immediate.

In the above proof both (2) and (3) imply $d(X) = d(Y) = k + 1$. Hence last part of the lemma follows if we set $A_1 = X \cap Y, A_2 = V - (X \cup Y), B_1 = X - Y, B_2 = Y - X$. □

We can now state a procedure to split off most edges in G. (The procedure is used in two proofs of Section 5.) Assume that every partition class P_j has $d(s, P_j) \leq d(s)/2$ in G. Starting with G, repeatedly choose an index i such that $d(s, P_i)$ is maximum and split off an allowed pair sx, sy with $x \in P_i$. The relation $d(s, P_j) \leq d(s)/2$ for all j is maintained throughout this procedure by the argument of Lemma 2.13. Lemmas 3.3 and 3.4 show that eventually either we get a complete allowed splitting or we get a graph that has a C_4-obstacle and $d(s) = 4$.

Lemma 3.5 *Suppose that the pair sx, sy is an allowed splitting and the graph $G_{x,y}$ contains a C_6-obstacle. Then G contains another allowed splitting sx, sy' (i.e., the split still involves edge sx) such that $G_{x,y'}$ contains no C_4-obstacle and no C_6-obstacle.*

Proof: Suppose there is a C_6-obstacle $A_1, B_1, C_1, A_2, B_2, C_2$ in $G_{x,y}$. Denote the unique neighbours of s in each of these sets by $a_1, b_1, c_1, a_2, b_2, c_2$. By the definition of a C_6-obstacle, there exist $a \neq b \neq c \neq a$ such that $a_1, a_2 \in P_a$, $b_1, b_2 \in P_b$, $c_1, c_2 \in P_c$.

We distinguish two cases corresponding to whether x and y belong to the same set or different sets of the obstacle. In the first case we may assume w.l.o.g. that $x, y \in A_1$. In the second case x and y must belong to sets that are consecutive in the C_6-obstacle, so we may assume w.l.o.g. that $x \in A_1$ and $y \in B_1$.

Case 1: $x, y \in A_1$.

At least one of c_1, b_2, say w.l.o.g. c_1, is not from the same partition class as x. Let $y' = c_1$. We claim that the pair sx, sy' is admissible in G. For suppose not and let Y be a maximal dangerous set containing x and y'. Since sx, sy is an admissible pair we have $y \notin Y$.

Lemma 2.8 shows Y contains at most 4 neighbours of s. Thus $Y \cup D \neq V$ for any set D of the obstacle. This implies Y and A_1 are crossing (since $y \notin Y$). It also implies Y contains any other set of the obstacle that it shares a vertex with (by Lemma 2.6). Since Y is connected (Lemma 2.2) and contains x and y', we conclude that $Y \cup A_1 = A_1 \cup B_1 \cup C_1 \cup D$ for $D = \emptyset, A_2$ or C_2. Now part (ii') of the definition of C_6-obstacle shows that in all cases for D, $d(A_1 \cup Y) \geq (k-1)/2 + (k-1)/2 + 5 = k + 4$. But applying (2) shows $(k+1) + (k+2) \geq d(Y) + d(A_1) \geq d(Y \cap A_1) + d(Y \cup A_1) \geq k + (k+4)$, a contradiction. Thus Y does not exist and sx, sy' is admissible.

Suppose $G_{x,y'}$ contains an obstacle. Call this the new obstacle, as opposed to the original C_6-obstacle. The new obstacle has sets A'_1, A'_2, B'_1, B'_2 and if it is a C_6-obstacle then it has sets C'_1, C'_2 as well.

Let D' be any set in the new obstacle, and let D be any set in the original obstacle except A_1 (so $d(D) = k$). In $G_{x,y'}$, neither D nor D' can contain 3 neighbours of s (a set in a C_4-obstacle contains ≤ 2 neighbours, if $d(s) = 6$). Hence $D \cup D' \neq V$. Furthermore Lemma 2.5 implies the two sets do not cross. So $D \subseteq D'$ if the two sets share a vertex (we cannot have $D' \subset D$ since that would make D crossing with another set of the new obstacle). We shall use this principle frequently.

By renaming if necessary, we can assume that $C_1 \subset A'_1$. The inclusion is proper since C_1 has no neighbour of s in $G_{x,y'}$.

We now show $A'_1 \cap A_1 \neq \emptyset$. Suppose the contrary. Then A'_1 has degree k in G. By Lemma 2.2 A'_1 induces a connected graph in G. Now it is easy to see (using the above principle) that we must have $A'_1 = X \cup C_1 \cup Y$, where X is either B_1 or empty, and Y is either empty or one of the three sets A_2, $A_2 \cup B_2$, $A_2 \cup B_2 \cup C_2$. In all cases (ii') of Definition 3.2 implies that A'_1 has degree at least $k+1$ in G. This is the desired contradiction.

We next show that $A_1 \cap A'_1$ contains no neighbour of s in $G_{x,y'}$. Since A'_1 induces a connected graph in G (Lemma 2.2) and has at most two neighbours of s in $G_{x,y'}$, our principle implies that $B_1 \subset A'_1$ and that $B_2 \cap A'_1 = \emptyset$. Applying (2) to the sets A_1 and A'_1 (of degree $k+2$ and k respectively in $G_{x,y'}$) we see that in $G_{x,y'}$ $A'_1 \cup A_1$ has degree at most $k+2$. Now it is easy to prove, by examining the neighbours of s in $G_{x,y'}$, that $A'_1 \cap C_2 = A'_1 \cap A_2 = \emptyset$. Hence $A'_1 \cup A_1 = A_1 \cup B_1 \cup C_1$. Thus in $G_{x,y'}$ we have $d(A'_1 - A_1) = k+1$. Then by (3),

$d(A_1 \cap A'_1, V - (A_1 \cup A'_1)) = 0$ holds in $G_{x,y'}$. This implies the desired conclusion.

Suppose first that the new obstacle is a C_4-obstacle. This implies two of the sets A'_2, B'_1, B'_2 contain exactly two neighbors of s. Lemma 2.7 shows these sets do not cross A_1. Now our principle shows at least one of these sets is the union of exactly two sets of the original obstacle. But this contradicts Lemma 2.4.

Now suppose that the new obstacle is a C_6-obstacle. Since the unique neighbour b_1 of s in A'_1 in the graph $G_{x,y'}$ belongs to P_b, it follows from Definition 3.2 and our principle that $A'_2 = B_2$. Thus $G_{x,y'} - s$ has a path from A'_1 to A'_2 that contains vertices in only one other new obstacle set (namely the new obstacle set containing A_2). This contradicts the fact that in a C_6-obstacle any path avoiding s from A'_1 to A'_2 goes through at least two other obstacle sets.

Case 2: $x \in A_1$ and $y \in B_1$.

As in Case 1 at least one of c_1, b_2 is not from the same partition class as x. In the first case we let $y' = c_1$ and in the second case we let $y' = b_2$. We claim that the pair sx, sy' is admissible in G. Suppose not and let Y be a maximal dangerous set containing x, y' in G. Since $d(A_1) = k$ it follows from Lemma 2.6 and Lemma 2.8 that $A_1 \subset Y$. Suppose first that $y' = c_1$. Since Y is connected by Lemma 2.2, we see as we did in Case 1 that either $B_1 \subset Y$ or $A_2 \cup B_2 \cup C_2 \subset Y$, both of which contradict Lemma 2.8. If $y' = b_2$, then the only case which does not lead to a contradiction as above is $Y = B_2 \cup C_2 \cup A_1$. But then $d(Y) = k + 2$, a contradiction. Hence the pair sx, sy' is admissible for splitting.

Suppose that $G_{x,y'}$ contains a new obstacle. Assume first that $y' = c_1$. As in Case 1 we let A'_1 be that set of the new obstacle such that $C_1 \subset A'_1$. The same argument as in Case 1 shows that $A'_1 \cap A_1 \neq \emptyset$. Lemma 2.5 (applied in $G_{x,y'}$) shows that $A_1 \subset A'_1$. Now as in Case 1 we can prove that $B_1 \subset A'_1$. But then A'_1 contains three neighbours of s in $G_{x,y'}$, a contradiction.

Assume finally that $y' = b_2$. Let A'_1 be that set of the new obstacle that contains B_2. Again we deduce that $A_1 \subset A'_1$. Using the fact that A'_1 induces a connected set in G and has at most 2 neighbours of s in $G_{x,y'}$, we get $A'_1 = B_2 \cup C_2 \cup A_1$. Thus A'_1 has precisely two neighbours of s. Thus the new obstacle is a C_4-obstacle A'_1, B'_1, A'_2, B'_2. In graph $G_{x,y'}$, B_1 contains two neighbours of s and has degree k. Hence Lemma 2.5 and Definition 3.1 show that w.l.o.g. $B'_1 = B_1$. Now it is easy to see that each of the sets $A'_1 \cup A'_2$ and $B'_1 \cup B'_2$ contain neighbours of s from two different sets of the partition $\mathcal{P}$, contradicting Definition 3.1. □

Theorem 3.6 *There exists a complete allowed splitting at vertex s in G if and only if*

(a) $d_i \leq d(s)/2$ for all $1 \leq i \leq r$,

(b) G contains neither a C_4-obstacle nor a C_6-obstacle.

Proof: (necessity) Suppose that there exists a complete allowed splitting off at vertex s in G. Since all the splittings are allowed, (a) is satisfied.

Suppose that G contains a C_4-obstacle. By (iii) and (iv) of Definition 3.1 the same partition remains a C_4-obstacle each time a split is executed. But then after executing all the splits the sets A_1, B_1 and B_2 contradict Lemma 2.3.

Now suppose that G contains a C_6-obstacle. Assume that a_1 is the unique neighbour of s in A_1 and suppose that the edge sa_1 was split off with sx. Then x cannot lie in A_2 because the splitting is allowed. Moreover, it can be neither in B_1 nor in C_2 since $A_1 \cup B_1$ and $A_1 \cup C_2$ are dangerous sets. Thus x is in either C_1 or B_2, say in C_1. Let us split off the pair sa_1, sx. Then $A_1 \cup B_1 \cup C_1, A_2, B_2, C_2$ is a C_4-obstacle, a contradiction by the above argument. Hence, if G contains a C_6-obstacle it cannot have a complete allowed splitting at s. □

Proof: (sufficiency) Assume that G satisfies (a) and (b). We use induction on $d(s)$ to show that there exists a complete allowed splitting at vertex s in G. If $d(s) = 2$ then the result clearly holds. If $d(s) = 4$ then the result follows from Lemma 3.4. Hence assume that $d(s) \geq 6$.

Without loss of generality assume d_1 is the maximum value d_i. Using Lemma 3.3, split off an allowed pair sx, sy where $x \in P_1$. $G_{x,y}$ satisfies condition (a) of Theorem 3.6 (as in the proof of Lemma 2.13). If $G_{x,y}$ contains no obstacle then by induction we are done.

Suppose $G_{x,y}$ contains a C_6-obstacle. It follows from Lemma 3.5 that we can replace sx, sy by another splitting sx, sy' such that $G_{x,y'}$ has no C_4-obstacle and no C_6-obstacle. Thus we are done by induction.

The remaining case is when $G_{x,y}$ contains a C_4-obstacle, say A_1, A_2, B_1, B_2. Then x and y are either in the same set or in two consecutive sets of the obstacle. Hence we may assume w.l.o.g. that either $x, y \in A_1$ (Case 1) or $x \in A_1$ and $y \in B_1$ (Case 2). In both cases we shall first show the following:

There is another allowed split sx', sy' that shares an edge, say sz, with sx, sy and has $x' \in P_j$ where $d_j = d_1$. Furthermore $z \in A_1$, and choosing vertex w' so $\{x', y'\} = \{w', z\}$, $w' \in A_2$. Also choose vertex w so $\{x, y\} = \{w, z\}$.

In Cases 1–2 below fix the graph as G.

Case 1: $x, y \in A_1$.

This means that in G, $d(A_1) = k + 2, d(A_2) = d(B_1) = d(B_2) = k$.

If A_2 contains a neighbour u of s not in P_1, then let $x' := x$ and $y' := u$. Otherwise, let x' be any neighbour of s in A_2 (x' exists by Lemma 2.3) and let $y' := y$. Note that either $x = x'$ or $y = y'$, and the two pairs sx, sy and sx', sy' are as desired.

Claim 3.7 *sx', sy' is admissible.*

Proof: If sx', sy' is not admissible then there exists a maximal dangerous set Y containing x' and y'. Y does not contain vertex w because sx, sy is admissible. Consequently $A_1 - Y \neq \emptyset$. $G[Y]$ is connected by Lemma 2.2. This implies that Y intersects $B_1 \cup B_2$ (say B_1) because there is no edge between A_1 and A_2. Lemma 2.6 shows that Y contains both A_2 and B_1.

We show that $Y \cap B_2 = \emptyset$. Otherwise Lemma 2.6 shows $B_2 \subseteq Y$. Then $d(s, Y) \geq (d(s) - 2)/2 + 1 + 1$, because $d(s, B_1 \cup B_2) = (d(s) - 2)/2, d(s, A_2) \geq 1, d(s, A_1 \cap Y) \geq 1$. This is a contradiction against Lemma 2.8.

Note that $d(s, A_1 \cup Y) \geq (d(s) - 2)/2 + 2 + 1 = d(s)/2 + 2$. Now Lemma 2.10 gives $(k + 2) + (k + 1) \geq d(A_1) + d(Y) \geq 2k + 2d(s, A_1 \cup Y) - d(s) \geq 2k + 4$. This contradiction shows sx', sy' is admissible. ♠

Case 2: $x \in A_1$ and $y \in B_1$.

This means that in G, $d(A_1) = d(A_2) = d(B_1) = d(B_2) = k$. If A_2 contains a neighbour u of s not in P_1, then let $x' := x$ and $y' := u$.

Otherwise, A_1 must contain a neighbour of s not in P_1 because this partition is not a C_4-obstacle in G. Thus in the graph $G_{x,y}$ all the neighbours of s in $B_1 \cup B_2$ belong to the same class P_j, $j \neq 1$. This implies that in G, $d_j \geq (d(s)-2)/2$ and $d_1 \leq (d(s)-2)/2 - 1 + 1$. Since d_1 was maximum it follows that $d_1 = d_j = (d(s)-2)/2$ in G and $y \notin P_j$. In this case let x' be any neighbour of s in B_2 (and thus $x' \in P_j$) and $y' := y$. Note that although $x' \notin P_1$ it belongs to a class P_j where d_j is also maximum. Now rename the sets A_1, A_2, B_1, B_2 as B_1, B_2, A_2, A_1 respectively. In both cases the two pairs sx, sy and sx', sy' are as desired.

Claim 3.8 *sx', sy' is admissible.*

Proof: If sx', sy' is not admissible then there exists a maximal dangerous set Y containing x' and y'. The vertex w does not belong to Y because the pair sx, sy was admissible. Thus $B_1 - Y \neq \emptyset$. Hence Lemma 2.6 shows $A_1 \cup A_2 \subseteq Y$.

$G[Y]$ is connected by Lemma 2.2. This implies that Y intersects $B_1 \cup B_2$ because there is no edge between A_1 and A_2. If Y intersects B_i then Lemma 2.6 shows Y contains B_i. Hence $i = 2$ and $B_2 \subseteq Y$. Then $d(s, Y) \geq d(s)/2 + 1$ because $d(s, A_1 \cup A_2) = d(s)/2$ and $d(s, B_2) \geq 1$ by Lemma 2.3. This contradicts Lemma 2.8. ♠

We have now found x', y' as desired. As shown previously, if $G_{x',y'}$ contains no obstacle or contains a C_6-obstacle then we are done. The remaining possibility is when $G_{x',y'}$ contains a C_4-obstacle. This cannot occur if the hypotheses of the theorem hold, because of the following lemma. (The lemma will also be used for our algorithm.) Let the C_4-obstacle in $G_{x',y'}$ be $A_1^*, A_2^*, B_1^*, B_2^*$, where w.l.o.g. A_1^* contains vertex z of the common edge sz of the two splittings.

Lemma 3.9 *After possibly interchanging B_1 and B_2, and/or B_1^* and B_2^*, the sets $A_1 \cap A_1^*$, B_1^*, A_2^*, B_2, A_2, B_1 form a C_6-obstacle in G.*

Proof:

Claim 3.10 *No two of the eight given obstacle sets cover V.*

Proof: Let C (C^*) be any set from the four sets A_1, A_2, B_1, B_2 $(A_1^*, A_2^*, B_1^*, B_2^*)$. Assume $C \cup C^* = V$. Then C (C^*) contains the other three sets from the second (first) obstacle. By Lemma 2.3, each of the eight obstacle sets contains a neighbour of s in G. Using this and property (iv) for both C_4-obstacles we get that $d(s, C - C^*) \geq (d(s)-2)/2 + 1 = d(s)/2$ and equality may hold only if $x', y' \in C^*$. Similarly, $d(s, C^* - C) \geq d(s)/2$ and equality may hold only if $x, y \in C$. However the two splits share an edge, so if equality holds in both relations then

$d(s, C \cap C^*) > 0$. Thus $d(s) > d(s, C - C^*) + d(s, C^* - C) \geq d(s)/2 + d(s)/2$, a contradiction. ♠

Case A: $d(A_1) = d(A_1^*) = k + 2$.

We show the sets of Lemma 3.9 have the properties of a C_6-obstacle using the following nine claims. Note that in Case A, $z \in A_1 \cap A_1^*$ and $w' \in A_2 \cap A_1^*$.

Claim 3.11 *A_1 and A_1^* are crossing sets.*

Proof: As just mentioned $A_1^* \cap A_1$ contains z so it is nonempty. $A_1 \cup A_1^* \neq V$ by Claim 3.10. $A_1^* \not\subseteq A_1$ because of vertex w'. Furthermore, $A_2 \subseteq A_1^*$. (This follows from Lemma 2.7 since $w' \in A_1^* \cap A_2$ and $z \in A_1^* - A_2$.) Since $d(s, A_1 \cup A_2) = (d(s) - 2)/2 + 2 = d(s, A_1^*) + d(s, A_2^*)$ we cannot also have $A_1 \subseteq A_1^*$ (since $d(s, A_2^*) > 0$). Hence A_1 and A_1^* are indeed crossing sets. ♠

By Lemma 2.2, $G[A_1^*]$ is connected. Thus A_1^* intersects $B_1 \cup B_2$. We also have that A_1 intersects $B_1^* \cup B_2^*$. To prove this note that Claim 3.11 shows $A_1 \cap (A_2^* \cup B_1^* \cup B_2^*) \neq \emptyset$. If $A_1 \cap A_2^* \neq \emptyset$ the desired conclusion follows by applying Lemma 2.2 as above. Now w.l.o.g. assume that A_1 intersects B_1^* and A_1^* intersects B_1.

Claim 3.12 *$B_1 \cup A_2 \subseteq A_1^*$, $B_1^* \cup A_2^* \subseteq A_1$ and $B_2 = B_2^*$.*

Proof: Since $w' \in A_2 \cap A_1^*$, Lemma 2.7 shows $B_1 \cup A_2 \subseteq A_1^*$. A_1^* cannot intersect B_2. (If it did, Lemma 2.7 implies $B_2 \subseteq A_1^*$. Then $A_1 \cup A_1^* = V$, contradicting Claim 3.10.) Hence $A_1^* - A_1 = B_1 \cup A_2$.

We now show that $A_1 - A_1^* = B_1^* \cup A_2^*$ (as claimed) by eliminating all other possiblities. Lemma 2.7 shows that A_1 does not cross any of the sets A_2^*, B_1^*, B_2^*. Recall that B_1^* intersects A_1, so $B_1^* \subseteq A_1 - A_1^*$. Suppose $A_1 - A_1^* = B_1^*$. Then $B_2 = A_2^* \cup B_2^*$ contradicting Lemma 2.4. Suppose $A_1 - A_1^* = B_1^* \cup B_2^*$. Then Lemma 2.2 shows $d(A_1 - A_1^*) \geq k + 3$. But then $d(s, A_1 \cap A_1^*) \geq 1$ and (3) give $k + 2 + k + 2 = d(A_1) + d(A_1^*) = d(A_1 - A_1^*) + d(A_1^* - A_1) + 2d(s, A_1 \cap A_1^*) \geq k + 3 + k + 2$, a contradiction. Claim 3.10 shows we cannot have $A_2^* \cup B_1^* \cup B_2^* \subset A_1$. We conclude that $A_1 - A_1^* = B_1^* \cup A_2^*$.

The third part of the claim follows from the other two. ♠

Note that Claim 3.12 suffices for Lemma 3.9 when there are only two partition classes, because it shows the lemma is vacuous: (B_1, B_2) is a nonconsecutive pair in A_1, A_2, B_1, B_2 so all neighbors of s in B_1 and B_2 are in the same partition class. But (A_1^*, B_2^*) is a consecutive pair in the other obstacle, so all neighbors of $B_1 \subseteq A_1^*$ and $B_2 \subseteq B_2^*$ are in different partition classes. We continue the analysis for the general case.

Claim 3.13 *$d(A_1 - A_1^*) = k + 1 = d(A_1^* - A_1)$, $d(s, A_1^* \cap A_1) = 1$ and $d(B_1, A_2) = (k-1)/2 = d(B_1^*, A_2^*)$.*

Proof: By Lemma 2.4 $d(A_1 - A_1^*)$ and $d(A_1^* - A_1)$ are at least $k + 1$, since they are both the disjoint union of two sets of degree k. Then (3) and the fact that $z \in A_1^*$ give $k + 2 + k + 2 =$

$d(A_1) + d(A_1^*) = d(A_1 - A_1^*) + d(A_1^* - A_1) + 2d(A_1^* \cap A_1, V + s - (A_1^* \cup A_1)) \geq k + 1 + k + 1 + 2$. Thus equality holds everywhere implying that $d(A_1 - A_1^*) = k + 1 = d(A_1^* - A_1)$, and $d(A_1^* \cap A_1, V + s - (A_1^* \cup A_1)) = 1$. In other words, $d(s, A_1^* \cap A_1) = 1$ and $d(A_1^* \cap A_1, B_2) = 0$.

The last relation $d(B_1, A_2) = (k-1)/2$ follows from $k + 1 = d(A_1^* - A_1) = d(B_1 \cup A_2) = d(B_1) + d(A_2) - 2d(B_1, A_2) = k + k - 2d(B_1, A_2)$. Similarly $d(B_1^*, A_2^*) = (k-1)/2$. ♠

Claim 3.14 $d(A_1^* \cup A_1) \geq k + 4$.

Proof: Obviously $d(s, A_1 \cup A_2) = d(s, B_1 \cup B_2) + 2$. Thus by Lemma 2.3, $d(s, A_1^* \cup A_1) = d(s, A_1 \cup A_2 \cup B_1) \geq d(s, A_1 \cup A_2) + 1 = d(s, B_1 \cup B_2) + 3 \geq d(s, B_2) + 4$. So $k = d(B_2) = d(A_1^* \cup A_1) - d(s, A_1^* \cup A_1) + d(s, B_2) \leq d(A_1^* \cup A_1) - 4$, as desired. ♠

Claim 3.15 $d(A_1 \cap A_1^*) = k$ *and* $d(A_1, A_1^*) = 0$.

Proof: Applying (2) for A_1 and A_1^* and using Claim 3.14 we have $k+2+k+2 = d(A_1)+d(A_1^*) = d(A_1 \cap A_1^*) + d(A_1 \cup A_1^*) + 2d(A_1, A_1^*) \geq k + (k+4)$. Thus equality holds everywhere and the claim follows. ♠

Claim 3.16 $d((A_1 \cap A_1^*) \cup B_1^*) = k + 1 = d((A_1 \cap A_1^*) \cup B_1)$.

Proof: Lemma 2.4 implies $d((A_1 \cap A_1^*) \cup B_1^*) \geq k+1$. Thus using Claim 3.13, $k+2 = d(A_1) = d((A_1 \cap A_1^*) \cup B_1^*) + d(A_2^*) - 2d(A_2^*, B_1^* \cup (A_1 \cap A_1^*)) \geq k + 1 + k - 2(k-1)/2 = k + 2$. This gives the first part of the claim. The second part is similar. ♠

Claim 3.17 $d(A_1 \cap A_1^*, B_1^*) = (k-1)/2 = d(A_1 \cap A_1^*, B_1)$ *and* $d(s, B_1^*) = 1 = d(s, B_1)$.

Proof: We prove both equalities for B_1^*. $k + k - 2d(A_1 \cap A_1^*, B_1^*) = d(A_1 \cap A_1^*) + d(B_1^*) - 2d(A_1 \cap A_1^*, B_1^*) = d((A_1 \cap A_1^*) \cup B_1^*) = k+1$ by Claim 3.16. The first part of the claim follows.

Now using Claim 3.13, $1 \leq d(s, B_1^*) \leq d(B_1^*) - d(A_1 \cap A_1^*, B_1^*) - d(A_2^*, B_1^*) = k - (k-1)/2 - (k-1)/2 = 1$. Thus $d(s, B_1^*) = 1$. ♠

By Claim 3.16 $d((A_1 \cap A_1^*) \cup B_1^*) = k+1$ so this set does not contain both x and y. It follows that $w \in A_2^*$. (Recall also $w' \in A_2$.)

Let the unique neighbour of s in B_1^* (B_1) belong to P_q (P_r).

Claim 3.18 $d(s, A_2) = 1 = d(s, A_2^*) = 1$.

Proof: After splitting off sx', sy', $B_1 \subseteq A_1^*$ and $S_r \cap B_1 \neq \emptyset$. Thus Definition 3.1(iii) shows $S_r \cap B_2^* = \emptyset$. Similarly, $S_q \cap B_2 = \emptyset$. By Lemma 2.3, $B_2 = B_2^*$ contains at least one neighbour of s, thus $B_1 \cup B_2$ intersects at least two classes of $\mathcal{P}$. This implies that after splitting off sx, sy in G, $S_q = (A_1 \cup A_2) \cap S$ and $d_q = (d(s) - 2)/2$. However, considering again $G_{x',y'}$, A_1^* cannot intersect S_q since $S_q \cap B_1^* \neq \emptyset$. Thus $d(s, A_2) = 1$. Similarly, $S \cap (A_1^* \cup A_2^*) = S_r$ and $d(s, A_2^*) = 1$. ♠

Claim 3.19 $d(A_2, B_2) = (k-1)/2 = d(A_2^*, B_2)$ *and* $d(s, B_2) = 1$.

Proof: Since $d(A_2, s) = 1$, $d(A_2, B_1) = (k-1)/2$ and $B_1 \cup B_2 \cup s$ contains all the neighbours of A_2, we see that $d(A_2, B_2) = (k-1)/2$. Similarly, $d(A_2^*, B_2) = (k-1)/2$. Then from $d(B_2) = k$ we obtain that $d(s, B_2) = 1$. ♠

To finish showing we have a C_6-obstacle we must prove (iv') holds. It follows from the Claims that $d(s) = 6$. In the proof of Claim 3.18, it was shown that $d_q = d_r = (d(s)-2)/2 = 2$ and $S \cap (A_2 \cup B_1^*) = S_q, S \cap (A_2^* \cup B_1) = S_r$. It was also shown that $B_2 \cap (S_q \cup S_r) = \emptyset$. Moreover since both splittings are allowed we have $(A_1 \cap A_1^*) \cap (S_q \cup S_r) = \emptyset$. It remains to show that if the unique neighbour of s in B_2 (in $A_1 \cap A_1^*$) belongs to P_d (P_e) of the partition $\mathcal{P}$ then $d = e$.

Note that $x \in P_r \cup P_e$. Hence A_2 contains a neighbor of s not in the same partition class as x. Hence Case 1 chooses $x' = x = z$. Since $x \in P_1$, $d_e = \max d_j = 2$. Hence $d = e$ and we are done. △

Case B: At least one of $d(A_1)$ and $d(A_1^*)$ is k.

Claim 3.20 *The partitions of the two obstacles coincide.*

Proof: All four sets of one of the obstacles have degree k in G. Claim 3.10 and Lemmas 2.4, 2.5 show that any set of degree k in the other obstacle equals one of the sets in the first obstacle. Since there are at least three such sets, the fourth set of both obstacles coincide too. ♠

The claim implies in particular that $d(A_1) = d(A_1^*) = k$. Recalling Case 2 we have $x \in A_1, y \in B_1$ and $w' \in A_2$. Now it is easy to see that

$(*)$ (A_1, A_2) and (B_1, B_2) are both (non)consecutive pairs in the second (first) C_4-obstacle.

In $G_{x,y}$ either $A_1 \cup A_2$ or $B_1 \cup B_2$ contains neighbours of s from only one class P_j of $\mathcal{P}$, by $(*)$ and Definition 3.1(iii). In $G_{x',y'}$ both sets of this union contain neighbors of s in P_j. (If the union is $A_1 \cup A_2$ note that $d(s, A_1), d(s, A_2) \geq 2$, since Lemma 2.3 shows each obstacle set in $G_{x',y'}$ contains a neighbour of s.) This contradicts Definition 3.1(iii) by $(*)$. We conclude that Case B cannot occur. This completes the proof of both Lemma 3.9 and Theorem 3.6. □

4 Increasing the edge-connectivity to an even number with partition constraints

Let $H = (V, E)$ be a graph with a partition $\mathcal{P} = \{P_1 \cup \ldots \cup P_r\}$, $r \geq 2$ of V. For the rest of this paper fix Φ to be the maximum of the following $r+1$ quantities:

$\alpha = \max\{\lceil \frac{\sum_{X\in\mathcal{F}}(k-d(X))}{2} \rceil : \mathcal{F}$ a subpartition of $V\}$;

$\beta_i = \max\{\sum_{Y\in\mathcal{F}_j}(k - d(Y)) : \mathcal{F}_j$ a subpartition of $P_i\}$ for $i = 1, 2, \ldots, r$.

Observe for any k of arbitrary parity, $OPT^k_{\mathcal{P}}(H) \geq \Phi$. In proof, clearly $OPT^k_{\mathcal{P}}(H) \geq OPT^k(H) = \alpha$. Also $OPT^k_{\mathcal{P}}(H) \geq \beta_i$ since we are not allowed to add an edge inside any P_i.

Lemma 4.1 *If k is even then $OPT^k_{\mathcal{P}}(H) = \Phi$.*

Proof: We give a constructive proof to show $OPT^k_{\mathcal{P}}(H) \leq \Phi$. Steps 1–2 of the proof are valid even when k is odd, and will be used in the next section.

Step 1. Execute the first two phases of Frank's algorithm (described in Section 2) with input H, k. Now there exists a subpartition $\mathcal{F}$ of V such that $d(s)/2 = OPT^k(H) = \lceil \frac{\sum_{X\in\mathcal{F}}(k-d(X))}{2} \rceil$. In particular $d(s)/2 \leq \alpha$ (actually equality holds).

Step 2. If there is some P_i for which $d(s, P_i) > d(s, V - P_i)$ then repeat the following three steps to make $d(s, P_i) = d(s, V - P_i)$.

Step 2a. For each edge su, $u \in P_i$ let X_u be a minimal critical set containing u.

Step 2b. If $X_u \not\subseteq P_i$ for some $u \in P_i$ then choose any $v \in X_u - P_i$ and replace the edge su by an edge sv. Lemma 2.11 shows the resulting graph still satisfies (1). If we now have $d(s, P_i) = d(s, V - P_i)$ then Step 2 is complete, so continue to Step 3. Otherwise (we still have $d(s, P_i) < d(s, V - P_i)$) go back to Step 2a.

Step 2c. At this point $X_u \subseteq P_i$ for every edge su, $u \in P_i$. Let $\mathcal{F}_i$ be the collection of maximal sets X_u. The same proof as [6, page 35] shows that $\mathcal{F}_i$ is a subpartition of P_i and $d(s, P_i) = \sum\{k - d(Y) : Y \in \mathcal{F}_i\}$. Now add $d(s, P_i) - d(s, V - P_i)$ edges from s to $V - P_i$ arbitrarily. This makes $d(s)/2 \leq \beta_i$.

Step 3. Now we have that in the current graph G, (1) holds and $d(s, P_j) \leq d(s, V - P_j)$ for $j = 1, 2, \ldots, r$. Hence it follows from Lemma 2.13 that we can obtain a complete allowed splitting off at s. This proves $OPT^k_{\mathcal{P}}(H) \leq d(s)/2 \leq \Phi$ as desired. $\square$

We close by noting that even when k is odd, Steps 1–2 make $\Phi = d(s)/2 \geq d(s, P_i)$ for all $i = 1, \ldots, r$. In proof, the argument above shows $d(s)/2 \leq \Phi$ for k odd. Step 1 sets $d(s)/2 = \alpha$ so the final graph has $d(s)/2 \geq \alpha$. The final graph has $d(s)/2 \geq d(s, P_i)$ for all i by Step 2. Also $d(s, P_i) \geq \beta_i$ since (1) holds. The desired relations follow.

5 Increasing the edge-connectivity to an odd number with partition constraints

This section treats the more complicated case when the target connectivity is an odd number. Let $H = (V, E)$ be a fixed graph, $k \geq 3$ be an odd positive integer and $\mathcal{P} = \{P_1, \ldots, P_r\}$ be a prescribed partition of V. Furthermore let $G = (V + s, E \cup F)$ be the graph obtained by executing Steps 1–2 of the algorithmic proof of Lemma 4.1.

We begin with a simple upper bound for partition-constrained augmentation. The proof generalizes the fact that the four-cycle can be augmented to a 3-edge-connected bipartite graph by duplicating three edges.

Lemma 5.1 *$OPT^k_{\mathcal{P}}(H) \leq \Phi + 1$.*

Proof: The remark at the end of Section 4 shows that every partition class P_i has $d(s, P_i) \leq d(s)/2$ in G. Now execute the procedure given after Lemma 3.4 to split off edges of G. As remarked, eventually either (i) we get a complete allowed splitting, or (ii) we get a C_4-obstacle in the (modified) graph G having $d(s) = 4$.

The proof of Lemma 4.1 and its discussion shows $\Phi \leq OPT^k_{\mathcal{P}}(H)$ and the original graph G has $d(s)/2 = \Phi$. Thus in case (i) we have $OPT^k_{\mathcal{P}}(H) = \Phi$. In case (ii) we get a sequence of $\Phi - 2$ allowed splittings that add a set of $\Phi - 2$ edges F' to G. Thus it suffices to prove that we can make the graph $H + F'$ k-edge-connected by adding three edges without violating the partition constraints.

In the rest of this argument fix G as the final graph having $d(s) = 4$. Lemma 3.4 shows G contains a C_4 obstacle A_1, A_2, B_1, B_2 such that each of the sets A_1, A_2, B_1, B_2, $A_1 \cup B_1$, $A_1 \cup B_2$ have degree $k - 1$ in $G - s$. Let us denote the unique neighbour of s in each of the 4 sets A_1, A_2, B_1, B_2 by a_1, a_2, b_1, b_2, respectively. Since Frank's algorithm makes $G - s$ k-edge-connected by splitting off two edges of G, it must add the edges a_1a_2 and b_1b_2.

We show that $G - s$ is $(k - 1)$-edge-connected. Suppose some $Z \subset V$ has degree at most $k - 2$ in $G - s$. We can assume $a_1 \in Z$ (if not replace Z by its complement). Then $a_2 \notin Z$ and Z contains precisely one of b_1, b_2, say b_2. Thus Z forms a crossing pair with $A_1 \cup B_1$. Note that b_1b_2 does not leave either of the sets $Z \cap (A_1 \cup B_1)$, $Z \cup A_1 \cup B_1$. Hence (2) shows $(k-2)+(k-1) \geq d(Z)+d(A_1 \cup B_1) \geq d(Z \cap (A_1 \cup B_1))+d(Z \cup A_1 \cup B_1) \geq (k-1)+(k-1)$, a contradiction. We conclude that $G - s$ is $(k - 1)$-edge-connected.

A degree $k-1$ set of $G-s$ contains exactly one of the vertices a_1, a_2 or exactly one of the vertices b_1, b_2 (by Frank's algorithm). Now it is easy to see that that adding the edges a_1b_1, b_1a_2, a_2b_2 makes $G - s$ k-edge-connected. □

It is possible that our graph $G = (V + s, E \cup F)$ contains a C_4- or C_6-obstacle and yet $OPT^k_{\mathcal{P}}(H) = \Phi$. This can occur when we can move certain edges incident to s so that a complete allowed splitting exists. This motivates our next two definitions, which identify the configurations in the given graph H that force $OPT^k_{\mathcal{P}}(H) > \Phi$ (this is proved in Theorem 5.8).

Definition 5.2 *Let X_1, X_2, Y_1, Y_2 be a partition of V with the following properties in H:*

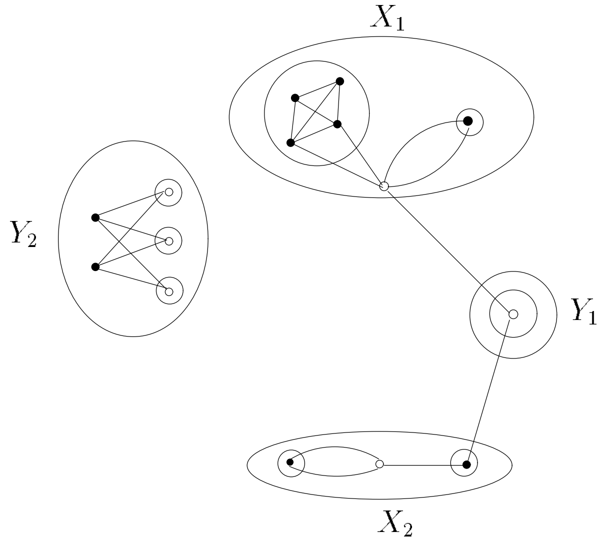

Figure 3: A graph $H = (V, E)$ with a partition $\mathcal{P} = \{P_1, P_2\}$ of V, where P_1 consists of the black vertices and P_2 consists of the white vertices. The sets X_1, X_2, Y_1, Y_2 form a C_4-configuration in H with respect to $k = 3$. The inner circles indicate the subpartitions $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}'_1, \mathcal{F}'_2$ as defined in Definition 5.2.

(i) $d(A) < k$ *for* $A = X_1, X_2, Y_1, Y_2$*;*

(ii) $d(A, B) = 0$ *for* $(A, B) = (X_1, X_2), (Y_1, Y_2)$*;*

(iii) There exist subpartitions $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}'_1, \mathcal{F}'_2$ *of* X_1, X_2, Y_1, Y_2 *respectively, such that for* A *ranging over* X_1, X_2, Y_1, Y_2 *and* $\mathcal{F}$ *the corresponding subpartition of* A*,* $k - d(A) = \sum_{U \in \mathcal{F}}(k - d(U))$*. Furthermore for some* $i \leq r$*,* P_i *contains every set of either* $\mathcal{F}_1 \cup \mathcal{F}_2$ *or* $\mathcal{F}'_1 \cup \mathcal{F}'_2$*.*

(iv) $(k - d(X_1)) + (k - d(X_2)) = (k - d(Y_1)) + (k - d(Y_2)) = \Phi$.

Such a partition is called a C$_4$-configuration *of* H*. (See Figure 3.)*

As with C_4-obstacles, k must be odd in a C_4-configuration.

Definition 5.3 *Let* $X_1 \cup X_2 \cup Y_1 \cup Y_2 \cup Z_1 \cup Z_2$ *be a partition of* V *with the following properties in* H*:*

(i') $d(A) = k - 1$ *for* $A = X_1, X_2, Y_1, Y_2, Z_1, Z_2$;

(ii') $d(A, B) = (k-1)/2$ *for* $(A, B) = (X_1, Y_1), (Y_1, Z_1), (Z_1, X_2), (X_2, Y_2), (Y_2, Z_2), (Z_2, X_1)$;

(iii') For three distinct partition classes P_a, P_b, P_c *there are six sets of degree* $k-1$*,* $X'_1, X'_2, Y'_1, Y'_2, Z'_1, Z'_2$ *contained in* $X_1 \cap P_a, X_2 \cap P_a, Y_1 \cap P_b, Y_2 \cap P_b, Z_1 \cap P_c, Z_2 \cap P_c$ *respectively.*

(iv') $\Phi = 3$.

Such a partition is called a C_6-configuration *of* H.

Lemma 5.4 *If H contains a C_4-configuration then it cannot be made k-edge-connected by adding Φ edges without violating the partition constraints.*

Proof: Suppose H contains a C_4-configuration X_1, X_2, Y_1, Y_2. Assume w.l.o.g. that $\bigcup_{U \in \mathcal{F}_1 \cup \mathcal{F}_2} U \subseteq P_i$ for some $i \leq r$. ($\mathcal{F}_1$ and $\mathcal{F}_2$ are as defined in Definition 5.2.) Suppose that we can add Φ edges to H such that the new graph H' becomes k-edge-connected and no new edge is added inside some $P_j \in \mathcal{P}$. Since $(k - d(X_1)) + (k - d(X_2)) = \Phi$ our assumption shows exactly one endvertex of each new edge is in $X_1 \cup X_2$. Hence no edge is added between X_1 and X_2. Furthermore Definition 5.2(iv) shows the sets X_1, Y_1, Y_2 all have degree k in H'. But this contradicts Lemma 2.3. $\square$

Lemma 5.5 *If H contains a C_6-configuration then it cannot be made k-edge-connected by adding $\Phi = 3$ edges without violating the partition constraints.*

Proof: Suppose that $X_1, X_2, Y_1, Y_2, Z_1, Z_2$ form a C_6-configuration in H and we can make H k-edge-connected by adding three edges. Then each of the sets $X_1, X_2, Y_1, Y_2, Z_1, Z_2$ will be incident with one new edge. Since we use only three new edges, none of which is added inside some class of $\mathcal{P}$, it is easy to see that one of the new edges goes between two consecutive sets, say w.l.o.g. between X_1 and Y_1. Then $d(X_1 \cup Y_1) = k - 1$ in the new graph, a contradiction. $\square$

For the next two lemmas recall that $G = (V + s, E \cup F)$ is the graph obtained after executing Steps 1 and 2 of the algorithmic proof of Lemma 4.1.

Lemma 5.6 *Suppose $A_1, B_1, C_1, A_2, B_2, C_2$ is a C_6-obstacle in G that is not a C_6-configuration in H. Then we can replace one edge incident to s by a new edge incident to s so that the resulting graph satisfies (1) and has a complete allowed splitting.*

Proof: Let $S = \{a_1, a_2,\ b_1, b_2,\ c_1, c_2\}$ and $a_i \in A_i$, $b_i \in B_i$, $c_i \in C_i$ for $i = 1, 2$. Let P_a, P_b, P_c be the distinct classes of $\mathcal{P}$ such that $a_1, a_2 \in P_a$, $b_1, b_2 \in P_b$ and $c_1, c_2 \in P_c$. Let $A_i' \subseteq A_i, B_i' \subseteq B_i, C_i' \subseteq C_i$, $i = 1, 2$ be minimal subsets of degree $k - 1$ in $G - s$. Since $d(s) = 6$ our remark at the end of Section 4 shows $\Phi = d(s)/2 = 3$. Now since the given partition is not a C_6-configuration we can assume w.l.o.g. that $A_1' \cap P_d \neq \emptyset$ for some $d \neq a$. Let $x \in P_d \cap A_1'$ and replace the edge sa_1 by the edge sx. Denote the new graph by G'. This graph satisfies (1) by Lemma 2.11. G' does not contain a C_6-obstacle since s has exactly one neighbour in P_a.

Suppose G' contains a C_4-obstacle $A_1^*, A_2^*, B_1^*, B_2^*$, where the labelling is chosen so that s has only neighbours from one $P_i \in \mathcal{P}$ in $A_1^* \cup A_2^*$. Then by the definition of a C_4-obstacle, s has three neighbours in P_i. Hence x belongs to both $A_1^* \cup A_2^*$ and $P_b \cup P_c$, say w.l.o.g. $x \in A_1^* \cap P_b$. By Lemma 2.5 none of the sets $A_1^*, A_2^*, B_1^*, B_2^*$ form a crossing pair with any of the sets $A_1, B_1, C_1, A_2, B_2, C_2$. There are no two sets $Z \in \{A_1, B_1, C_1, A_2, B_2, C_2\}$ and $W \in \{A_1^*, A_2^*, B_1^*, B_2^*\}$ such that $Z \cup W = V$ since Z contains only one neighbour of s. This implies that $A_1^* \cup A_2^* = A_1 \cup B_1 \cup B_2$. But Lemma 2.4 implies this is impossible since all five sets have degree k. Hence G' contains no C_4-obstacle.

Now Theorem 3.6 shows G' has a complete allowed splitting at s. $\Box$

We say that a subpartition $\mathcal{F}$ of V is *irreducible* if every $X \in \mathcal{F}$ has $d(X) < k$.

Lemma 5.7 *Suppose A_1, B_1, A_2, B_2 is a C_4-obstacle in G that is not a C_4-configuration in H. Then either we can replace one edge incident to s by a new edge, or we can replace two edges incident to s by two new edges, so that the resulting graph satisfies (1) and has no C_4-obstacle.*

Proof: Choose the labelling of A_1, B_1, A_2, B_2 so all neighbours of s in $A_1 \cup A_2$ belong to one P_i. Let $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_1', \mathcal{F}_2'$ be arbitrary irreducible subpartitions of A_1, A_2, B_1, B_2 respectively such that in H, $k - d(A_1) = \sum_{U \in \mathcal{F}_1}(k - d(U))$, $k - d(A_2) = \sum_{U \in \mathcal{F}_2}(k - d(U))$, $k - d(B_1) = \sum_{W \in \mathcal{F}_1'}(k - d(W))$ and $k - d(B_2) = \sum_{W \in \mathcal{F}_2'}(k - d(W))$. Our remark at the end of Section 4 shows that in G, $d(s)/2 = \Phi$. Now since A_1, A_2, B_1, B_2 is not a C_4-configuration in H we must have that $\bigcup_{U \in \mathcal{F}_1 \cup \mathcal{F}_2} U$ contains a vertex $x \notin P_i$, say w.l.o.g. $x \in U \subseteq A_1$, $U \in \mathcal{F}_1$. Since $\mathcal{F}_1$ is irreducible s has a neighbour $u \in U \cap P_i$. Replace the edge su by the new edge sx. If all neighbours of s in $B_1 \cup B_2$ belong to some P_j, then since A_1, A_2, B_1, B_2 do not form a C_4-configuration in H it follows again that $\bigcup_{W \in \mathcal{F}_1' \cup \mathcal{F}_2'} W$ contains a vertex $y \notin P_j$, say w.l.o.g. $y \in W \subseteq B_1$, $W \in \mathcal{F}_1'$. The irreducibility of $\mathcal{F}_1'$ shows s has a neighbour $v \in W \cap P_j$. Replace the edge sv by the new edge sy.

Denote the graph resulting from these edge replacements by G'. G' satisfies (1) by Lemma 2.11. We will show G' contains no C_4-obstacle. (It may contain a C_6-obstacle.) Suppose G' contains a C_4-obstacle $A_1^*, A_2^*, B_1^*, B_2^*$ where the labelling is chosen so all neighbours of s in $A_1^* \cup A_2^*$ belong to one class of $\mathcal{P}$. An argument similar to ones in Section 3 shows the two obstacle partitions coincide. (Specifically Lemma 2.5 implies no two obstacle sets cross. No two obstacle sets cover V since one would violate Lemma 2.8. Lemma 2.4 completes the argument.)

Observe that $A_1^* \cup A_2^*$ is not the set $A_1 \cup A_2$ or $B_1 \cup B_2$, by our edge replacements. Thus

$$A_1^* \cup A_2^* = A_p \cup B_q$$

for some indices p, q. We claim that $d(s, A_p),\ d(s, B_q) > 1$. For suppose $d(s, A_p) = 1$ (the argument for $d(s, B_q) = 1$ is similar). Write $\overline{p} = 3 - p, \overline{q} = 3 - q$. Then $d(s, A_{\overline{p}}) = d(s)/2 - 1$, and so $d(s, B_{\overline{q}}) = 1$. The definition of the two C_4-obstacles shows no edge of H leaves $A_p \cup B_{\overline{q}}$. (An edge leaving A_p goes to $A_{\overline{p}} \cup B_{\overline{q}}$ by the second obstacle and to $B_q \cup B_{\overline{q}}$ by the first obstacle. Similarly for an edge leaving $B_{\overline{q}}$.) Thus $d(A_p \cup B_{\overline{q}}) = 2$, contradicting $k \geq 3$.

All neighbours of s in $A_1^* \cup A_2^*$ are in the same partition class. Thus if $p = 1$ then $d(s, A_1) = 1$ (by our edge replacement). If $p = 2$ then $q = 1$ and $d(s, B_q) = 1$ (since no edge was replaced in A_2). Both cases give a contradiction. We conclude the C_4-obstacle does not exist. $\Box$

The following result holds for k of arbitrary parity. It subsumes Lemma 4.1.

Theorem 5.8 *Let $k \geq 2$ and let $H = (V, E)$ be a graph with a partition $\mathcal{P} = \{P_1 \cup \ldots \cup P_r\}$, $r \geq 2$ of V. Then $OPT^k_{\mathcal{P}}(H) = \Phi$ unless H contains a C_4-configuration or a C_6-configuration, in which case $OPT^k_{\mathcal{P}} = \Phi + 1$.*

Proof: We give a constructive proof. (The same outline is followed by our efficient algorithm.) Start by performing Steps 1 and 2 of the procedure of Section 4, and let G be the resulting graph. As remarked at the end of Section 4 G has $d(s)/2 = \Phi$, and also $d(s, P_i) \leq d(s)/2$ for all $1 \leq i \leq r$. Now execute the procedure given after Lemma 3.4 to split off edges of G. As remarked, eventually we obtain either a complete allowed splitting or a graph with $d(s) = 4$ containing a C_4-obstacle A_1, A_2, B_1, B_2. In the first alternative $OPT^k_{\mathcal{P}}(H) = \Phi$. The rest of the proof concerns the second alternative. We distinguish two cases depending on how the new edges were added to G during the sequence of splittings.

Case I: At least one split added an edge inside one of the sets A_1, A_2, B_1, B_2.

Say w.l.o.g. that an edge was added inside A_1 when we split off the pair sx, sy. Undoing this split gives a graph G' that has $d(s) = 6$ and still satisfies (1). If neither x nor y belongs to a partition P_i with $d(s, P_i)$ maximum in G' then it is easy to see that G' does not contain an obstacle. Thus Theorem 3.6 shows G' has a complete allowed splitting and $OPT^k_{\mathcal{P}}(H) = \Phi$.

In the opposite case G', x and y have all the properties of G, x and y in Case 1 and Case A of the proof of Theorem 3.6. Following Case 1 we define another allowed splitting sx', sy' for which there are two possibilities. The first possibility is that splitting off sx', sy' gives a graph with no obstacle. Since this graph has $d(s) = 4$ Lemma 3.4 shows the allowed splitting can be completed. Thus $OPT^k_{\mathcal{P}}(H) = \Phi$.

Lemma 3.9 shows that the second possibility is that graph G' contains a C_6-obstacle. Suppose first that $G' \neq G$, i.e. $d(s) \geq 8$ in G. Let G' be the result of executing a split in the graph G''. Lemma 3.5 shows G'' has a splitting that results in a graph G^* where $d(s) = 6$ and there is no C_4-obstacle and no C_6-obstacle. Theorem 3.6 shows we can find a complete allowed splitting in G^*. Thus $OPT^k_{\mathcal{P}}(H) = \Phi$.

The remaining case is when $G' = G$, so G has a C_6-obstacle. Lemma 5.6 shows there are two possibilities. The first possibility is that we can repair this C_6-obstacle by moving one edge incident with s, to get a graph with a complete allowed splitting. This shows $OPT^k_{\mathcal{P}}(H) = \Phi$. The second possibility is that H has a C_6-configuration. In this case it follows from Lemmas 5.5 and 5.1 that $OPT^k_{\mathcal{P}}(H) = \Phi + 1$.

Case II: All splits added edges between different sets of the partition A_1, A_2, B_1, B_2.

In this case A_1, A_2, B_1, B_2 is a C_4-obstacle in G. Lemma 5.7 shows there are two possiblities. The first possibility is that A_1, A_2, B_1, B_2 is a C_4-configuration, in which case Lemmas 5.4 and 5.1 show $OPT^k_{\mathcal{P}}(H) = \Phi + 1$. The second possiblity is that we can move at most two edges incident with s so that the resulting graph G' has no C_4-obstacle. If G' has no C_6-obstacle either, then Theorem 3.6 shows $OPT^k_{\mathcal{P}}(H) = \Phi$. If G' has a C_6-obstacle then the theorem holds as shown above. $\square$

Corollary 5.9 *Let $k \geq 2$ and let $H = (V, E)$ be a graph with a partition $\mathcal{P} = \{P_1 \cup \ldots \cup P_r\}$, $r \geq 2$ of V. If $OPT^k(H) \geq 2k+1$, then $OPT^k_{\mathcal{P}}(H) = \Phi$.*

Proof: If H has a C_6-configuration then $OPT^k(H) = 3$ so the corollary is vacuous. If H contains a C_4-configuration then Definition 5.2(iv) shows $OPT^k(H) \leq 2k$. $\square$

Corollary 5.10 *Let $H = (V, E)$ be a bipartite graph with bipartition $V = X \cup Y$ and let $\mathcal{P} = \{X, Y\}$.*

- *Let $\alpha = \max\{\lceil \frac{\sum_{X \in \mathcal{F}}(k-d(X))}{2} \rceil : \mathcal{F}$ a subpartition of $V\}$.*
- *Let $\beta'_1 = \sum_{v \in X} \max\{0, k - d(v)\}$.*
- *Let $\beta'_2 = \sum_{v \in Y} \max\{0, k - d(v)\}$.*
- *Let $\Phi' = \max\{\alpha, \beta'_1, \beta'_2\}$.*

Then $OPT^k_{\mathcal{P}}(H) = \Phi'$ unless k is odd and H contains a C_4-configuration, in which case $OPT^k_{\mathcal{P}}(H) = \Phi' + 1$. $\square$

If $r = |V|$ we obtain the min-max equality of the unconstrained problem due to Cai and Sun.

Corollary 5.11 *[3] Let $H = (V, E)$ be an undirected graph and $k \geq 2$.*

- *Let $\alpha = \max\{\lceil \frac{\sum_{X \in \mathcal{F}}(k-d(X))}{2} \rceil : \mathcal{F}$ a subpartition of $V\}$.*

Then $OPT^k(H) = \alpha$. $\square$

6 Efficient algorithms

This section presents a strongly-polynomial algorithm for edge connectivity augmentation with partition constraints. The time bound is $O(n(m + n \log n) \log n)$, the same as the best-known bound for the problem without partition constraints. Throughout this section n and m denote the number of vertices and edges in the given graph, respectively. Each edge xy has a given integer capacity, which in our notation equals $d(x, y)$. We represent splits by providing the edges sx, sy and an integral *multiplicity* γ, which indicates that the split is to be performed γ times (producing γ new copies of the edge xy).

Although our main concern is graphs that have arbitrary capacity functions, we also state time bounds for the special case of unit capacity graphs. In these graphs each edge has capacity one. Parallel edges are allowed, but each copy of an edge contributes one to the number of edges m. For unit capacity graphs our algorithm runs in time $O(nm \log n)$, the best-known bound for the unconstrained problem on these graphs. Throughout this section time bounds that apply to unit capacity graphs are explicitly designated as such; time bounds without such an explicit designation apply to arbitrary capacitated graphs.

Say the split sx, sy *joins* vertices x and y. Throughout this section it is convenient to define

$$h = (k-1)/2.$$

6.1 Basic facts

Assume $G = (V+s, E)$ has $d(s)$ even and satisfies (1). Consider a set $X \subseteq V$. G/X denotes the graph G with X contracted to a single vertex. A split sx, sy in G corresponds to a split in G/X in the obvious way. Set X is *contractible* if any collection of splits in G that is admissible in G/X is admissible in G. (Recall a collection of splits is admissible if executing the splits results in a graph satisfying (1).)

We will generalize the following well-known fact first proved by Mader [15].

Proposition 6.1 *A critical set is contractible.*

Proof: Consider a critical set C and a collection of splits in G that is admissible in G/C. Let G' be the result of executing the splits in G. We must show that any nonempty set $X \subset V$ satisfies $d'(X) \geq k$ (as usual d' denotes the degree function in G').

Case 1: $X \subseteq C$.

No split joins two vertices of C, since $d(C) = k$. Thus the degree of any subset of C is preserved by the splitting, so $d'(X) \geq k$.

Case 2: $C \subseteq X$ or $X \cap C = \emptyset$.

Since the splitting is admissible in G/C, $d'(X) \geq k$.

Case 3. Neither Case 1 or Case 2 applies.

By (3),

$$d'(C) + d'(X) \geq d'(C-X) + d'(X-C).$$

Note that both sets $C - X$ and $X - C$ are nonempty. The analysis of Case 1 shows $d'(C-X) \geq k = d'(C)$. Case 2 shows $d'(X-C) \geq k$. Substituting these relations in the displayed inequality gives the desired conclusion $d'(X) \geq k$. □

To generalize Proposition 6.1 define a *k-bisection* of a set $C \subseteq V$ to be a partition of C into exactly two sets of degree k.

Lemma 6.2 *A dangerous set having no k-bisection is contractible.*

Proof: The argument is similar to Proposition 6.1. Consider a set C with $d(C) = k + 1$, and a collection of splits in G that is admissible in G/C. Taking G' and d' as in Proposition 6.1, we must show that any nonempty set $X \subset V$ satisfies $d'(X) \geq k$.

If X satisfies Case 1 or Case 2 of Proposition 6.1, we have $d'(X) \geq k$. (For Case 1 we use the fact that no split joins vertices of C since $d(C) = k + 1$.) If neither Case 1 or Case 2 applies then $C - X$ and $X \cap C$ are both nonempty. Since C has no k-bisection at least one of these sets has degree $\geq k + 1 = d'(C)$. If $d'(C - X) \geq d'(C)$ we argue as in Case 3 of Proposition 6.1. This leaves only the following case:

Case 4. $d'(C \cap X) \geq d'(C)$.

First suppose $C \cup X \neq V$. By (2),

$$d'(C) + d'(X) \geq d'(C \cap X) + d'(X \cup C).$$

Case 2 shows $d'(X \cup C) \geq k$. Substituting this and the inequality for Case 4 gives $d'(X) \geq k$.

Next suppose $C \cup X = V$. Applying Lemma 2.8 in G/C to C shows $d'(s, X) \geq d'(s, X - C) \geq d'(s, C) \geq d'(s, C - X)$. Thus $d'(X) \geq d'(C - X)$. Case 1 shows $d'(C - X) \geq k$ so $d'(X) \geq k$ as desired. □

Lemma 6.3 *Suppose $d(C) = k + 1$ and C has a k-bisection. Let the sets of the k-bisection be A and B. Then*

(a) k is an odd integer.

(b) A and B are both maximal degree k subsets of C, and C has no other maximal degree k subsets.

(c) Sets A and B form the unique minimum cut of graph $G[C]$. This cut has value $d(A, B) = h$.

Proof: (a) Applying (2) shows $2k = d(A)+d(B) = d(C)+2d(A, B)$. This implies $d(C) = k+1$ is even.

(b) Let M be a maximal subset of C having degree k. Assume M is distinct from A and B. We will deduce a contradiction, thereby proving (b).

M is not a subset of A or B. Since $d(M) = k$, $M \subset C$. Thus M is crossing with at least one of the sets A, B, say A. Since A and M are crossing critical sets, their union is critical by (2). This contradicts the maximality of M.

(c) Consider any cut of $G[C]$, say a partition of C into sets X and Y. Applying (2), $2k \leq d(X) + d(Y) = d(X \cup Y) + 2d(X, Y)$. Since $d(X \cup Y) = k + 1$ this implies $d(X, Y) \geq (k - 1)/2 = h$. Equality holds throughout for the cut A, B. Furthermore (b) shows that A, B is the unique k-bisection of C, so any other cut X, Y has $2k < d(X)+d(Y)$, whence $d(X, Y) > h$. This proves (c). □

We conclude this section by mentioning two simple properties of splittings that will be useful. As before let G satisfy (1) with $d(s)$ even. The first property is that any vertex x in a dangerous set has $d(s, x) \leq d(s)/2$ (equivalently, $d(s, x) \leq d(s, V - x)$). This follows from Lemma 2.8.

For the second property suppose that every vertex x has $d(s, x) \leq d(s)/2$ and furthermore, there is a unique complete splitting at s such that no split joins a vertex to itself. Then this splitting is admissible. To show this recall that a complete admissible splitting exists (Theorem 2.12(a)). If it contains a split sx, sx it also contains a split sy, sz with $y, z \neq x$ (since $d(s, x) \leq d(s)/2$). Replacing these two splits by sx, sy and sx, sz results in another admissible splitting. Performing this replacement as many times as possible results in the unique complete splitting with no split joining a vertex to itself, so this splitting is admissible.

6.2 The splitting algorithm for even k

Consider the problem of finding a complete admissible splitting at s. We present an algorithm that runs in time $O(n(m + n \log n) \log n_s)$. Here n_s is the number of neighbours of s, $n_s \leq n$. Our algorithm is a modification of the algorithm of Nagamochi et.al. [18], using contraction as the main operation. The main advantage of our approach is that it can be extended to find a complete allowed splitting for odd k. (Both algorithms can be so extended for even k.) In addition there is a slight efficiency advantage: The algorithm of [18] finds a complete splitting with $O(n_s \log n_s)$ distinct splits. Our splitting contains $O(n_s)$ distinct splits, the asymptotically optimum number. This implies fewer distinct edges get added to the graph. It improves the space bound, from $O(m + n_s \log n_s)$ to $O(m)$. Also while the time bound of [18] for k even is the same as our bound, for k odd it is our bound with $\log n_s$ changed to $\log n$ (so in some cases our algorithm is slightly faster). This section presents our splitting algorithm for even k.

We first summarize the algorithm of Nagamochi et.al. [18] (see also [17]). It is based on two routines. The first routine, C-SPLIT, finds a complete splitting at s. This splitting need not be admissible. However it has the following property. C-SPLIT is called with a subpartition of V whose sets contain all neighbours of s. No split found by C-SPLIT joins two vertices in the same partition set.

The second routine, HOOK-UP, is called with a complete splitting. It identifies a maximal subset of these given splits that is admissible. It executes these splits on the graph (so (1) is preserved). It also returns a family $\mathcal{Y}$ of disjoint dangerous sets, such that each given split that was not executed joins two vertices in the same dangerous set of $\mathcal{Y}$. (In terms of our representation of splits using multiplicities, the maximal admissible subset of splits is an assignment of a new multiplicity γ' to each given split sx, sy of multiplicity γ, with $\gamma' \leq \gamma$. Increasing any value γ' destroys admissiblity. If $\gamma' < \gamma$ then x and y are in the same set of $\mathcal{Y}$.) HOOK-UP runs in time $O(n(m + n \log n))$. (The time is $O(nm)$ on a unit capacity graph.)

The algorithm of [18] for even k is as follows. The algorithm is a loop that terminates when $d(s) = 0$, i.e., vertex s has been completely split off. Each iteration of the loop starts by using C-SPLIT to find a complete splitting, with no split joining two vertices in the same set of $\mathcal{Y}$. (For the first iteration $\mathcal{Y}$ is initialized to the family of singleton sets $\{x\}$ where x ranges over all neighbours of s. After the first iteration, $\mathcal{Y}$ was set by HOOK-UP in the previous iteration.) HOOK-UP is called to execute a maximal admissible subset of these splits, and to update $\mathcal{Y}$ for these splits. If $d(s) = 0$ then a complete admissible splitting at s has been executed, so the algorithm halts.

It is proved in [18] that in each iteration the size of the family $\mathcal{Y}$ is halved. This implies there are $O(\log n_s)$ iterations. Each iteration uses time $O(n(m + n \log n))$, so the total time is $O(n(m + n \log n) \log n_s)$.

We now present our modified splitting algorithm for even k. Our approach is to always contract the sets of $\mathcal{Y}$. Instead of C-SPLIT we use the following algorithm E-SPLIT to find a complete splitting. This algorithm seems easier to modify for the case of odd k and the case of allowed splittings. E-SPLIT outputs a complete splitting at s, with no split joining a vertex to itself (this specification differs slightly from C-SPLIT). Let $x_i, i = 1, \ldots, p$ be the distinct neighbours of s, indexed so that $d(s, x_1)$ equals the maximum degree $d(s, x_i)$. E-SPLIT is always called with

$$d(s, x_1) \leq \sum_{i>1} d(s, x_i).$$

E-SPLIT maintains d_i as the value of $d(s, x_i)$ if the splits output so far were to be executed. So initially $d_i = d(s, x_i)$ for $i = 1, \ldots, p$. E-SPLIT maintains the preceding inequality throughout its execution, i.e.,

$$d_1 \leq \sum_{i>1} d_i. \tag{7}$$

Procedure E-SPLIT

Step 1. While (7) holds with strict inequality, repeat the following Step 1a:

Step 1a. Choose distinct indices $i, i' > 1$ with both $d_i, d_{i'} > 0$. Output split $sx_i, sx_{i'}$ with multiplicity $\gamma = \min\{d_i, d_{i'}, (\sum_{i>1} d_i - d_1)/2\}$. Decrease d_i and $d_{i'}$ by γ.

Step 2. (At this point (7) holds with equality.) Output splits sx_1, sx_i with multiplicity d_i, for all indices $i > 1$ that have positive d_i.

We show that E-SPLIT works correctly, i.e., it outputs a complete splitting at s, with no split joining a vertex to itself. First observe that $d_1 = \max\{d_i : i \geq 1\}$ throughout the algorithm. This follows since Step 1a only decreases values d_i with $i > 1$. Also observe (7) holds throughout the algorithm, by the definition of γ and the termination condition of Step 1a.

Now consider Step 1. First observe there are at least two indices $i > 1$ with d_i positive, since (7) holds with strict inequality and every d_i, $i > 1$ is no larger than d_1.

Next observe that Step 1 eventually halts. This follows because the sum $\sum_{i>1} d_i$ has the same parity as d_1 when E-SPLIT starts (since $d(s)$ is even). Each split maintains this relation, since it decreases the sum by an even integer. Thus eventually (7) holds with equality and Step 1 completes.

Step 2 works correctly since (7) holds with equality.

To analyze the efficiency of E-SPLIT, first observe that it outputs at most p distinct splits. This follows because with one exception, every split (in Step 1 or Step 2) decreases some d_i, $i > 1$ to zero. The exception (which may not exist) is the unique split in Step 1 that achieves equality in (7).

Now it is easy to see that E-SPLIT uses total time $O(p)$.

Our splitting algorithm works by finding splits in a contracted graph H. We execute these splits in both the contracted graph H and the original graph G. (So both G and H change throughout the algorithm. The vertex set of G is always $V + s$.) The following invariant holds throughout the algorithm:

Graph H can be obtained from G by contracting a collection of disjoint dangerous subsets of V. Both G and H satisfy (1).

The invariant will guarantee correctness of the algorithm.

We use the following auxiliary routine to translate a split from H to G. It is called with two disjoint sets $X, Y \subset V$ and an integer γ. X and Y have been contracted to vertices in H, and the split sX, sY with multiplicity γ is admissible in H.

Procedure TRANSLATE(X, Y, γ)
Repeat Step 1 until $\gamma = 0$.

Step 1. Choose a vertex x of G corresponding to X: If X is a vertex of G let $x = X$. Otherwise choose a vertex $x \in X$ that is a neighbour of s in G. Similarly choose a vertex y of G corresponding to Y. Execute split sx, sy in G with multiplicity $\gamma_0 = \min\{d(s,x), d(s,y), \gamma\}$ (here d is the degree function in G). Decrease γ by γ_0.

Assuming the invariant holds, TRANSLATE executes a sequence of splits that is admissible in G. This follows because any dangerous set is contractible when k is even (by Proposition 6.1, Lemma 6.2 and Lemma 6.3(a)). The final G obtained by TRANSLATE corresponds to H with split sX, sY executed (so the invariant is preserved).

Now we present our splitting algorithm for even k. We assume the given graph G satisfies (1) with $d(s)$ even. In addition we assume that every vertex x satisfies $d(s,x) \le d(s)/2$. This inequality holds automatically in many applications of splitting, e.g., Frank's algorithm of Section 2. If not, we can enforce the inequality by deleting all but $d(s) - d(x)$ edges sx for the vertex x satisfying $d(s,x) > d(s)/2$. (To prove this note that in a complete admissible splitting at most $d(s) - d(x)$ copies of edge sx are split off with an edge sy, $y \ne x$, so the remaining copies are in splits sx, sx which do not affect the connectivity.)

The splitting algorithm works simultaneously on graphs G and H that satisfy the invariant stated above. The algorithm executes a complete admissible splitting at s on the original graph G. The algorithm starts with H initialized to the given graph G.

Procedure EVEN-SPLIT
Repeat Steps 1–3 until $d(s) = 0$.

Step 1. Call E-SPLIT to find a complete splitting at s in graph H, with no split joining a vertex to itself.

Step 2. Call HOOK-UP to execute a maximal admissible subset of these splits in H, and to return the corresponding family of dangerous sets $\mathcal{Y}$. For each split that was executed in H call TRANSLATE to execute it in G.

Step 3. In H contract each set $Y \in \mathcal{Y}$ to a single vertex.

To show EVEN-SPLIT is correct first observe that Step 1 works correctly because (7) holds whenever E-SPLIT is called. To prove (7) recall that it holds by assumption when the algorithm starts and E-SPLIT is called for the first time. In subsequent iterations Step 3 ensures that every vertex of H adjacent to s forms a dangerous set. This implies (7) (by the first remark at the end of Section 6.1).

Next observe that the invariant is preserved by all steps: The splits executed by HOOK-UP and TRANSLATE preserve the invariant (by the discussion of TRANSLATE). The contractions done in Step 3 clearly preserve the invariant.

The invariant implies that if EVEN-SPLIT halts, it has executed a complete admissible splitting on the original graph G. So we need only prove termination. We do this by showing there are $O(\log n_s)$ iterations in EVEN-SPLIT.

Consider graph H after Step 2. Any remaining neighbour x of s was in an unexecuted split sx, sy, with x and y now belonging to the same set of $\mathcal{Y}$. This set gets contracted in Step 3. Now let s_i denote the number of distinct neighbours of s at the start of the ith iteration. Then

$$s_{i+1} \leq s_i/2.$$

Since $s_1 = n_s$, there are $O(\log n_s)$ iterations.

Next we show the total number of distinct splits executed in Step 2 in all iterations is $O(n_s)$, for both G and H. This fact for graph H is important for reasons of efficiency: Each distinct split executed in Step 2 can add a new edge to H, increasing space and the time for subsequent computations. The fact for graph G is a measure of quality of the complete splitting produced by EVEN-SPLIT.

To prove the fact for H, Step 1 (of EVEN-SPLIT) gives a splitting with at most s_i distinct splits, by our discussion of E-SPLIT. So clearly Step 2 executes at most s_i distinct splits in H. The inequality displayed above shows the sum of all quantities s_i is $O(s_1) = O(n_s)$.

To prove the fact for G it suffices to show a total of at most n_s more splits are done in G than in H. This follows because for a given split sX, sY in H, each execution of Step 1 of TRANSLATE excepting the last decreases a value $d(s, x)$ to 0.

Now we can show the time for the entire algorithm EVEN-SPLIT is $O(n(m+n \log n) \log n_s)$. Graphs G and H both have $O(m + n_s) = O(m)$ edges. Each call to HOOK-UP takes time $O(n(m + n \log n))$. The remaining processing in EVEN-SPLIT takes time $O(m)$ in each iteration. There are $O(\log n_s)$ iterations, so the desired time bound follows. It is easy to show that for unit capacity graphs EVEN-SPLIT runs in time $O(nm \log n_s)$. The argument is the same as above, with the observation that G and H always have $O(m)$ edges because each split decreases the number of edges.

6.3 The splitting algorithm for odd k

We turn to our splitting algorithm for k an odd integer. The algorithm of [18] is based on the (strong) successive augmentation property [4]: It first executes splits to achieve connectivity $k - 1$, and then augments the connectivity of the resulting graph to k. Our algorithm is a modification of the algorithm for k even, again based on contraction. We present the algorithm by specifying the changes made to the k even algorithm. The resulting algorithm will apply to both even and odd k.

The main change in the algorithm is to avoid creating splits that are obviously wrong. Call a split sx, sy *foolish* if $\{x\}$ and $\{y\}$ are critical sets and $d(x,y) = h$. A foolish split is inadmissible, since $d(\{x,y\}) = d(x) + d(y) - 2d(x,y) = 2k - 2h = k+1$. We modify E-SPLIT so it creates at most C foolish splits, for some constant C. (We do not take care to minimize the value of C. Also the value of C will change when we modify the algorithm to do allowed splitting.)

We start with two observations about foolish splits. First, a vertex x is in at most two foolish splits. This follows because $\{x\}$ must be critical and x must be a neighbour of s, i.e., $d(x) = k = 2h+1$ and $d(s,x) \geq 1$. Thus $d(x,y) = h$ for at most two vertices y.

Second, in a complete splitting a vertex x is in at most one foolish split. This is obvious if $d(s,x) = 1$. If $d(s,x) > 1$ then the above calculation shows we can have $d(x,y) = h$ for at most one vertex y.

Now we present procedure G-SPLIT. Like E-SPLIT, it outputs a complete splitting with no split joining a vertex to itself. G-SPLIT is the same as E-SPLIT with one change. Step 1a chooses indices i, i' by executing the following Choice Step:

Choice Step. Choose an index $i > 1$ with $d_i > 0$. Let F be the set of indices $f > 1$ with sx_i, sx_f a foolish split. (Note $|F| \leq 2$). Choose an index $i' \neq i, 1$ with $d_{i'} > 0$ and, if possible, $i' \notin F$. If $i' \in F$ and no splits have been output yet in Step 1, redefine i and i' to be the two indices of F.

Note that if k is even there are no foolish splits, so G-SPLIT is the same as E-SPLIT. It is not immediately obvious that the last statement of the Choice Step is well-defined or correct. We now prove the algorithm works as desired. Assume that G-SPLIT (like E-SPLIT) is called with values d_i corresponding to vertices in a graph H that satisfies (1).

Lemma 6.4 *(a) G-SPLIT outputs a complete splitting, with no split joining a vertex to itself.*
(b) G-SPLIT outputs at most $C = 2$ foolish splits.
(c) The first split output by G-SPLIT is not foolish.

Proof: Regarding part (a), by definition G-SPLIT never outputs a split joining a vertex to itself. Now consider Step 2 of G-SPLIT. It outputs at most one foolish split, since all its splits involve vertex x_1. Suppose no splits are output in Step 1. This means G-SPLIT starts with equality in (7). Step 2 outputs the unique complete admissible splitting of H with no split joining a vertex to itself (by the second remark at the end of Section 6.1). Clearly (c) holds in this case.

To complete the proof we analyze Step 1 of G-SPLIT. For (b) it suffices to show Step 1 outputs at most one foolish split. We consider two cases.

Case 1: For the first value of i chosen in Step 1, $i' \notin F$.

This certainly implies part (c). Furthermore it ensures that the last statement of the Choice Step never redefines i and i'. Now suppose at some point Step 1 outputs a foolish split. Right before this occurs, at most three vertices x_j have $j > 1$ and $d_j > 0$ (specifically the vertices x_i and $x_{i'}$ in the foolish split, and at most one other vertex x_f with $f \in F$). Step 1 can output at most one foolish split involving only these three vertices (since a given vertex is in at most one foolish split that gets output). Thus (b) holds, as well as (a).

Case 2: Case 1 does not apply.

Let I be the set of all indices that are larger than 1 with d_i initially positive. A complete admissible splitting exists. Recall (7) holds with strict inequality when Step 1 is executed. This implies there is an admissible split $sx_i, sx_{i'}$ with $i, i' \in I$.

Suppose $|I| = 2$. Obviously the first value of i gives the only split that is possible. This split is not foolish since it is admissible. Thus Case 1 applies.

Now suppose $|I| \geq 3$. The first value of i gives a set with $F = I - \{i\}$. Thus $|I| \leq 3$ and so $|I| = 3$. Thus the unique admissible split formed from indices in I comes from the two indices in F. The last statement of the Choice Step outputs this split. This finishes Step 1 (again since a complete admissible splitting exists). Thus (a) – (c) hold. $\square$

G-SPLIT outputs at most p distinct splits, since it is a special case of E-SPLIT. It is easy to see that G-SPLIT runs in time $O(m + n)$.

Now we present the splitting algorithm GENERAL-SPLIT. This procedure constructs a complete admissible splitting for arbitrary k. As before we assume the given graph G satisfies (1) with $d(s)$ even, and every vertex x satisfies $d(s, x) \leq d(s)/2$. GENERAL-SPLIT is derived from EVEN-SPLIT by making two changes. Step 1 of GENERAL-SPLIT calls G-SPLIT rather than E-SPLIT. Step 3, which does the contractions, is replaced by the following.

Contraction Step. In H do the following for each set $Y \in \mathcal{Y}$: If Y is critical, contract Y. Otherwise ($d(Y) = k + 1$) test if Y has a k-bisection. If it does not, contract Y. If it does, say into sets A and B, contract A and contract B.

We make several remarks on the Contraction Step. If k is even there are no k-bisections. Hence GENERAL-SPLIT becomes identical to EVEN-SPLIT. If Y has a k-bisection, after contracting A and B a split sA, sB is foolish. And if sx, sy is a foolish split giving a set $Y = \{x, y\}$, the Contraction Step does not change the graph when it processes Y.

The Contraction Step tests if Y has a k-bisection by using the algorithm of [16] to find a minimum cut of $H[Y]$. Lemma 6.3(c) shows this cut is the k-bisection, if one exists. It is a simple matter to check if the cut is a k-bisection.

We proceed to analyze GENERAL-SPLIT. To prove correctness we first state the new invariant:

Graph H can be obtained from G by contracting a collection of disjoint subsets of V, each of which is either critical or a dangerous set having no k-bisection. Both G and H satisfy (1).

The TRANSLATE routine preserves the new invariant. This follows by the argument of Section 6.2 with one change: Any set contracted in H is contractible in G. This follows from Proposition 6.1 and Lemma 6.2.

Now we can use the same argument as for EVEN-SPLIT to show that in Step 1 of GENERAL-SPLIT (7) holds, and also GENERAL-SPLIT preserves its invariant. For the first argument note that the Contraction Step still ensures after the first iteration, every neighbour of s is in a dangerous set. For the second argument note that the Contraction Step clearly preserves the invariant.

We must also prove that GENERAL-SPLIT terminates. We do this by showing that in H, every iteration of GENERAL-SPLIT either decreases the number of neighbours of s or decreases the number of vertices. To do this recall the first split output by G-SPLIT is not foolish (Lemma 6.4(c)). Let this split be sx, sy with multiplicity γ.

First suppose HOOK-UP splits off γ pairs of edges sx, sy. We show this decreases the number of neighbours of s in H. This is certainly true if $\gamma = \min\{d(s,x), d(s,y)\}$. It is easy to see that when the remaining possibility for γ holds, sx, sy is the last (as well as the first) split output by Step 1 of G-SPLIT. Thus after executing this split equality holds in (7), and the remaining splits are the only ones possible in a complete admissible splitting at s. Hence these splits are all admissible. So this iteration of GENERAL-SPLIT executes a complete splitting, resulting in no remaining neigbors of s.

On the other hand suppose HOOK-UP splits off fewer than γ pairs of edges sx, sy. HOOK-UP returns a set $Y \in \mathcal{Y}$ containing both x and y, and $Y \neq \{x, y\}$ since the split is not foolish. Hence the Contraction Step decreases the number of vertices of H when it processes set Y. In summary we have shown that GENERAL-SPLIT always terminates.

Now we prove the time bound. The key step is to prove there are $O(\log n_s)$ iterations in GENERAL-SPLIT.

We first introduce some notation. Let H_i be the graph H at the start of the ith iteration of GENERAL-SPLIT. (Our argument will not refer to the corresponding graph G.) For example H_1 is the given graph. The degree function in H_i is denoted d_i. Suppose v is a vertex in H_i. For $j \geq i$, $\overline{v}$ denotes the vertex in graph H_j that contains v. When we use this notation the value of j will be explicitly stated or clear from the context. For instance $d_j(\overline{v})$ denotes the degree in H_j of vertex $\overline{v}$ in H_j.

We classify the contractions made in Contraction Steps as follows. Call a contraction *good* if it reduces the number of vertices that contain a neighbour of s in H_1. In other words let X be a set that gets contracted in graph H_i. In a good contraction X contains distinct vertices $\overline{v}$ and $\overline{w}$ (in H_i) where both v and w are neighbours of s in H_1. ($\overline{v}$ and $\overline{w}$ need not be neighbours of s in H_i.) A contraction that is not good is *bad*. It is easy to see that a contraction is good if $X \in \mathcal{Y}$. So in a bad contraction X is one of the two sets of a k-bisection.

In the following discussion assume the number of iterations is at least four. Thus graph H_i exists for $1 \leq i \leq 4$.

Lemma 6.5 *Let v be a neighbour of s in H_1. At least one of the following alternatives holds for some index i, $1 \leq i \leq 3$:*

(i) $\overline{v}$ is not a neighbour of s in H_4;

(ii) $\overline{v}$ is in a foolish split created in H_i;

(iii) $\overline{v}$ is in a good contraction done in H_i;

(iv) w, a neighbour of s in H_i with $d_i(\overline{v}, w) = h$, is in a good contraction done in H_j, for some j in $i < j \leq 3$.

Proof: We assume none of alternatives (i) – (iv) hold and derive a contradiction.

Since (i) fails, $\overline{v}$ is a neighbour of s in H_4. Thus for $1 \leq i \leq 3$, $\overline{v}$ is in a split created in H_i but not executed by HOOK-UP. For $1 \leq i \leq 3$ let $s\overline{v}, sw_i$ be such a split. (So w_i is a vertex of H_i.) We will show that in H_4 vertices $\overline{v}, \overline{w}_1, \overline{w}_2$ and $\overline{w}_3$ are distinct, and in addition satisfy inequality (8). This will lead to the desired contradiction.

Since (iii) fails, split $s\overline{v}, sw_i$ results in a bad contraction. This implies

$$d_{i+1}(\overline{v}) = k.$$

Also Lemma 6.3(c) shows

$$d_{i+1}(\overline{v}, \overline{w}_i) = h.$$

Again since (iii) fails, $\overline{v} \neq \overline{w}_i$ in H_4 for $1 \leq i \leq 3$. This plus the preceding equation implies that for $i < j \leq 4$,

$$d_j(\overline{v}, \overline{w}_i) \geq h. \tag{8}$$

Next we show that in H_4, $\overline{w}_i \neq \overline{w}_j$ for any indices i, j with $1 \leq i < j \leq 3$. First we show $\overline{w}_i \neq w_j$ in H_j. As noted above, $d_j(\overline{v}) = k$. Since (iv) fails, $d_j(\overline{w}_i) = k$. By (8), $d_j(\overline{v}, \overline{w}_i) \geq h$. If equality holds then $\overline{w}_i \neq w_j$ because (ii) fails. If strict inequality holds then $\overline{w}_i \neq w_j$ because $d_{j+1}(\overline{v}, \overline{w}_i) > h = d_{j+1}(\overline{v}, \overline{w}_j)$. We conclude that $\overline{w}_i \neq w_j$ in H_j.

Thus to have $\overline{w}_i = \overline{w}_j$ in H_4 a contraction must merge the distinct vertices $\overline{w}_i$ and $\overline{w}_j$ in some graph H_r, $r > j$. But since (iv) fails this never occurs. Hence $\overline{w}_i \neq \overline{w}_j$ in H_4.

Now the distinctness of vertices $\overline{v}, \overline{w}_1, \overline{w}_2$ and $\overline{w}_3$ in H_4 and inequality (8) in H_4 imply

$$d_4(\overline{v}) \geq d_4(\overline{v}, \overline{w}_1) + d_4(\overline{v}, \overline{w}_2) + d_4(\overline{v}, \overline{w}_3) + d_4(\overline{v}, s) \geq 3h + 1 > 2h + 1 = k.$$

(Note that $h \geq 1$ since $k \geq 3$.) But as noted above, $d_4(\overline{v}) = k$. This contradiction proves the lemma. □

Let s_i denote the number of neighbours of s in H_i. Let $C' = 3C$. We will use the lemma to show

$$s_4 - C' \leq (5/6)(s_1 - C'). \tag{9}$$

In proof, let S be the set of all neighbours of s in H_1. Define the following quantities corresponding to the four cases of Lemma 6.5:

g is the number of vertices of S that are in a good contraction in at least one H_i, $1 \leq i \leq 3$.

n is the number of vertices of S, not counted in g, that are not neighbours of s in H_4. (This conflicts with the usual usage of n but will not cause confusion.)

f is the number of vertices of S, not counted in g or n, in a foolish split in some H_i, $1 \leq i \leq 3$.

r is the number of vertices of S not counted in g, n or f.

By definition $s_1 = g + n + f + r$.

First observe that the vertices counted in g belong to at most $g/2$ distinct vertices in H_4. This follows because a good contraction merges at least two vertices of S.

The vertices counted in n are no longer neighbours of s in H_4. Thus we deduce

$$s_4 \leq g/2 + f + r.$$

Next we show

$$r \leq 2g.$$

Lemma 6.5(iv) applies to any vertex v that is counted in r. Thus some vertex w counted in g has $d_4(\overline{v}, \overline{w}) \geq h$. Any contraction results in a vertex of degree at most $k+1$. Hence $d_4(\overline{w}) \leq k+1$. This implies $\overline{w}$ has $d_4(\overline{v}, \overline{w}) \geq h$ for at most four vertices v (since $5h \geq 2h+3 = k+2$). Thus $r \leq 4(g/2) = 2g$ as desired.

Finally recall that each iteration creates at most C foolish splits. Hence three iterations create at most $3C = C'$ foolish splits. Thus

$$f \leq C'.$$

Combining the above inequalities shows

$s_4 - C' \leq g/2 + f + r - C' \leq g/2 + (5/6)(f+r) + C'/6 + g/3 - C' = (5/6)(g+r+f-C') \leq (5/6)(s_1 - C')$.

This proves (9).

For convenience set $s_i = 0$ if i is larger than the number of iterations, so i ranges over all integers ≥ 1.

Lemma 6.6 *(a) There are $O(\log n_s)$ iterations of GENERAL-SPLIT.*
(b) $\sum_i s_i = O(n_s)$.

Proof: We can apply (9) to an arbitrary graph H_1. Hence for any index i,

$$s_{i+3} - C' \leq (5/6)(s_i - C').$$

This shows there are $O(\log n_s)$ iterations where s has more than C' neighbours. It also implies the sum of all values s_i that are larger than C' is $O(n_s)$.

To complete the proof (of both parts) it suffices to show that $s_i \leq C'$ in $O(1)$ iterations i. To do this we strengthen our proof of termination of GENERAL-SPLIT. We claim that every iteration of GENERAL-SPLIT either decreases s_i or increases the number of foolish splits that exist in H. This claim completes the proof: It implies that when $s_i \leq C'$, s_i decreases after at most $2C'$ iterations (since a vertex is in at most two foolish splits).

To prove the claim, consider an iteration that does not decrease s_i. A split su, sv that is foolish at the start of the iteration is foolish in the next iteration. In proof, since the Contraction Step does not merge any neighbours of s the next graph has $\overline{u} \neq \overline{v}$, $d(\overline{u}) = d(\overline{v}) = k$, and $d(\overline{u}, \overline{v}) \geq h$. The last inequality actually holds with equality, $d(\overline{u}, \overline{v}) = h$ (otherwise $2k = d(\overline{u}) + d(\overline{v}) = d(\overline{u} \cup \overline{v}) + 2d(\overline{u}, \overline{v}) \geq k + 2(h+1) = 2k+1$, a contradiction). So $s\overline{u}, s\overline{v}$ is a foolish split.

Now consider the first split sx, sy output by G-SPLIT. Recall this split is not foolish, and our proof of termination of GENERAL-SPLIT shows HOOK-UP returns a set $Y \in \mathcal{Y}$ containing x and y. In the Contraction Step Y has a k-bisection (since the Contraction Step does not merge neighbours of s). So the contractions make $s\overline{x}, s\overline{y}$ a new foolish split. $\square$

As with EVEN-SPLIT, the total number of distinct splits executed in Step 2 in all iterations is $O(n_s)$, for both G and H. This follows from the same argument as EVEN-SPLIT, using Lemma 6.6(b) in the analysis of graph H.

Now we show the total time for GENERAL-SPLIT is $O(n(m+n\log n)\log n_s)$. The previous paragraph shows each graph H_i has $O(m+n_s) = O(m)$ edges. Each call to HOOK-UP takes time $O(n(m+n\log n))$. Testing each graph $H[Y]$ for a k-bisection involves running the edge connectivity algorithm of [16]. It uses time $O(n(m+n\log n))$ on a graph of n vertices and m edges. In one Contraction Step, the total number of vertices in all graphs $H[Y]$ is at most n and each $H[Y]$ has at most m edges. This implies the total time for one Contraction Step is $O(n(m+n\log n))$. (A tighter analysis shows all k-bisection tests in the entire algorithm use total time $O(n(m+n\log n))$.) The remaining processing in GENERAL-SPLIT takes time $O(m)$ in each iteration. Lemma 6.6(a) shows there are $O(\log n_s)$ iterations, so the desired time bound follows.

The same analysis shows that the time on unit capacity graphs is $O(nm\log n_s)$. We use the fact that the edge connectivity algorithm of [16] runs in time $O(nm)$ on these graphs. In summary we have proved the following refinement of [18].

Theorem 6.7 *Procedure GENERAL-SPLIT finds a complete admissible splitting at a vertex s in time $O(n(m+n\log n)\log n_s)$ and space $O(m)$. The splitting contains $O(n_s)$ distinct splits. The time on unit capacity graphs is $O(nm\log n_s)$.* □

6.4 Complete splittings respecting partitions and degree sequences

The main task in adapting the algorithm of Sections 6.3 to allowed splitting is modifying G-SPLIT to only create splits that join two different partition classes. This section solves this problem.

Before discussing the problem recall that the *degree sequence* of a graph consists of the integers $d(x)$, where x ranges over every vertex. Hakimi [9] proved that given integers d_i, $i = 1,\ldots,p$ are the degree sequence of a multigraph with no self-loops if and only if $\sum_i d_i$ is even and $d_1 \le \sum_{i>1} d_i$, where the indexing is chosen so $d_1 = \max\{d_i : 1 \le i \le p\}$.

Recall that E-SPLIT finds a complete splitting at s, with no split joining a vertex to itself. In the given (multi)graph let s have neighbours $x_i, i = 1,\ldots,p$ and let d_i denote $d(s,x_i)$. The splitting found by E-SPLIT exists precisely when d_i is the degree sequence of a multigraph with no self-loops. Now it is clear that our algorithm E-SPLIT gives an alternate proof of Hakimi's theorem. (The necessity of Hakimi's condition is clear.) The algorithms of this section for complete splitting give other theorems on degree sequences, which we state with the algorithm.

We now state the problem solved in this section more precisely. In the graph G that is given for allowed splitting, let the partition be $\mathcal{P} = \{P_i : 1 \le i \le r\}$. Recall that our routines E-SPLIT and G-SPLIT are executed on a contraction H of G. For each neighbour x of s in H and each partition class P_i, $1 \le i \le r$, let $d_i(s,x)$ denote the total number of edges joining s to a vertex v of G that is in both vertex x and P_i. Thus $d(s,x) = \sum_i d_i(s,x)$. The task of E-SPLIT, when we compute allowed splittings, is to output a complete splitting that can be written as splits sx_i, sy_j, where x and y are distinct vertices of H, i and j are indices of distinct partition classes, and the total multiplicity of all splits involving x_i is $d_i(s,x)$. (Note that in the initial iteration when $H = G$, a given vertex x has $d_i(s,x) > 0$ for at most one index i.) The task of G-SPLIT for allowed splittings is the same as E-SPLIT with the additional requirement of not creating too many foolish splits (as in Lemma 6.4).

6.4.1 Even target connectivity k

We begin with the algorithm used when the target k is even and the partition has exactly two classes. This routine is of special interest because it is used for bipartite connectivity augmentation. Also it is a subroutine of our algorithm for general partitions.

Let us restate the problem to be solved in a convenient notation. We are given nonnegative integers d_{i1}, d_{i2}, $i = 1, \ldots, p$, such that

$$\sum_i d_{i1} = \sum_i d_{i2}. \tag{10}$$

(Note that in our splitting algorithm the index i corresponds to vertex x_i, d_{i1} and d_{i2} correspond to $d_1(s, x_i)$ and $d_2(s, x_i)$ respectively.) The task of the problem is to output a *complete pairing*, defined as a collection of ordered pairs of indices ii' where $i \neq i'$, with corresponding nonnegative multiplicities $\gamma_{ii'}$, such that $d_{i1} = \sum_{i'} \gamma_{ii'}$ and $d_{i'} = \sum_i \gamma_{ii'}$. Choose indices so that $d_{11} + d_{12} = \max\{d_{i1} + d_{i2} : 1 \leq i \leq p\}$. We will show a complete pairing exists if

$$d_{11} + d_{12} \leq \sum_{i>1} d_{i1} + d_{i2}. \tag{11}$$

First note the following interpretation in terms of degree sequences. Recall a party graph is a complete bipartite graph with a perfect matching deleted. (Every person at the party talks to every member of the opposite sex except his or her spouse.) We generalize this slightly. For index i ranging from 1 to p, say that d_{i1}, d_{i2} is the *degree sequence of a party graph* if there is a bipartite multigraph with vertices x_{i1} on one side, x_{i2} on the other side, no edge $x_{i1}x_{i2}$, and $d(x_{ij}) = d_{ij}$ for $j = 1, 2$. We prove d_{i1}, d_{i2} is the degree sequence of a party graph if and only if (10) and (11) hold, where the indexing is as described above. The necessity of these conditions is clear.

Our algorithm for two partition classes and even k is the following procedure BE-SPLIT. It maintains (10) and (11) throughout its execution.

Procedure BE-SPLIT

Step 1. While (11) holds with strict inequality, repeat the following Step 1a:

Step 1a. Choose distinct indices $i, i' > 1$ with $d_{i1}, d_{i'2} > 0$. Output pair ii' with multiplicity $\gamma = \min\{d_{i1}, d_{i'2}, (\sum_{i>1}(d_{i1} + d_{i2}) - (d_{11} + d_{12}))/2\}$. Decrease d_{i1} and $d_{i'2}$ by γ.

Step 2. (At this point (11) holds with equality.) Output pairs $1i$ with multiplicity d_{i2} and $i1$ with multiplicity d_{i1}, for all indices $i > 1$ that have positive d_{i2} or d_{i1} respectively.

We show that BE-SPLIT outputs a complete pairing. Consider Step 1. First we show the indices in Step 1a exist.

We claim some index $i > 1$ has d_{i1} positive. In proof (10) shows $\sum_{i>1} d_{i2} = \sum_i d_{i1} - d_{12}$. Substituting in (11) with strict inequality gives

$$d_{11} + d_{12} < 2\sum_{i>1} d_{i1} + d_{11} - d_{12},$$

which implies the claim. Similarly some $i' > 1$ has $d_{i'2}$ positive. If these indices i and i' are distinct they form the desired pair for Step 1a. Suppose they are identical, i.e., some $i > 1$ has $d_{i1}, d_{i2} > 0$. An index $i' > 1$ distinct from i has $d_{i'1} + d_{i'2}$ positive, since (11) holds with strict inequality and $d_{i1} + d_{i2} \le d_{11} + d_{12}$ ($d_{11} + d_{12} = \max\{d_{i1} + d_{i2} : i \ge 1\}$ throughout the algorithm). Thus either ii' or $i'i$ form the desired pair.

Next observe that Step 1 eventually halts. This follows because the two sides of (11) have the same parity (by (10)). Each pair output maintains this relation. So eventually (11) holds with equality and Step 1 halts.

Finally we show Step 2 works correctly, i.e., the pairs that are output exist and account for all remaining values d_{i1}, d_{i2}. When we have equality in (11) the inequality displayed above becomes an equality and implies $d_{12} = \sum_{i>1} d_{i1}$. Similarly $d_{11} = \sum_{i>1} d_{i2}$. These two relations show the pairs in Step 2 exist and exhaust all remaining values d_{i1}, d_{i2}, i.e., a complete pairing has been output.

Next we analyze the efficiency of BE-SPLIT. It outputs $O(p)$ distinct pairs. The argument is the same as for E-SPLIT: Excluding the last pair output in Step 1, every pair decreases some d_{i1} or d_{i2} to zero.

BE-SPLIT can be implemented to use total time $O(p)$. In Step 1 we maintain a list of all indices $i > 1$ having positive d_{i1}, and a similar list for d_{i2}. An execution of Step 1a chooses the first index from both lists; if they are the equal it also chooses the second index from one of the lists.

This concludes the discussion of BE-SPLIT (we have also proved the theorem on party graph degree sequences). We turn to the problem for general partitions (and even target k). As before we first restate the problem in a convenient notation. We are given nonnegative integers d_{ij} for $i = 1, \ldots, p$ and $j = 1, \ldots, q$. In our splitting algorithm i corresponds to vertex x_i, j corresponds to partition class j, and d_{ij} corresponds to $d_j(s, x_i)$. The given values have these properties for an integer D:

$$\sum_{i,j} d_{ij} = 2D;$$

$$\sum_{j} d_{ij} \le D \text{ for } i = 1, \ldots, p;$$

$$\sum_{i} d_{ij} \le D \text{ for } j = 1, \ldots, q.$$

(Note $\sum_{i,j} d_{ij}$ equals $\sum_{i=1}^{p} \sum_{j=1}^{q} d_{ij}$.) The task of the problem is to output a *complete pairing*, defined as a collection of pairs of ordered pairs $ij, i'j'$ where $i \ne i'$ and $j \ne j'$, with corresponding nonnegative multiplicities $\gamma_{ij,i'j'} = \gamma_{i'j',ij}$, such that for every $i = 1, \ldots, p$ and $j = 1, \ldots, q$, $d_{ij} = \sum_{i',j'} \gamma_{ij,i'j'}$.

In terms of degree sequences we prove the following. Let indices i and i' range over $1 \leq i, i' \leq p$, j and j' range over $1 \leq j, j' \leq q$. Say that d_{ij} is the *degree sequence of a p, q-party graph* if there is a multigraph with vertices x_{ij}, edges $x_{ij}x_{i'j'}$ where $i \neq i'$ and $j \neq j'$, and degrees $d(x_{ij}) = d_{ij}$. We prove d_{ij} is the degree sequence of a p, q-party graph if and only if the three conditions displayed above hold. The necessity of these conditions is clear.

Our algorithm incorporates an additional goal of limiting the number of pairs ii' that have pairs $ij, i'j'$ which are output. It is convenient to state the algorithm using the following notation. Choose indices so that $\max\{\sum_j d_{ij} : 1 \leq i \leq p\} \cup \{\sum_i d_{ij} : 1 \leq j \leq q\}$ equals either $\sum_j d_{1j}$ or $\sum_i d_{i1}$. In the first case define $\delta_i = 1$, $\delta_j = 0$. In the second case define $\delta_i = 0$, $\delta_j = 1$. Thus in both cases,

$$\max\{\textstyle\sum_j d_{ij} : 1 \leq i \leq p\} \cup \{\sum_i d_{ij} : 1 \leq j \leq q\} = \sum_{i=\delta_i \text{ or } j=\delta_j} d_{ij}.$$

(In the summation on the right the indices range over all pairs i, j such that either $i = \delta_i$ and $1 \leq j \leq q$ or $j = \delta_j$ and $1 \leq i \leq p$.) We assume that $d_{ij} = 0$ if i or j is zero.

Our algorithm for an arbitrary partition and even k is the following procedure PE-SPLIT. It maintains the following relations throughout its execution:

$$\sum_{i,j} d_{ij} \text{ is even;} \tag{12}$$

$$\sum_{i=\delta_i \text{ or } j=\delta_j} d_{ij} \leq \sum_{i \neq \delta_i, j \neq \delta_j} d_{ij}. \tag{13}$$

Notice that for the given values d_{ij} these conditions are equivalent to those assumed for the input.

Procedure PE-SPLIT

Step 1. While (13) holds with strict inequality, repeat the following Steps 1a–1b:

Step 1a. Choose distinct indices $i, i' > \delta_i$ such that there are distinct indices $j, j' > \delta_j$ with $d_{ij}, d_{i'j'} > 0$.

Step 1b. While (13) holds with strict inequality and there are distinct indices $j, j' > \delta_j$ such that $d_{ij}, d_{i'j'} > 0$, output the pair $ij, i'j'$ with multiplicity $\gamma = \min\{d_{ij}, d_{i'j'}, (\sum_{i \neq \delta_i, j \neq \delta_j} d_{ij} - \sum_{i=\delta_i \text{ or } j=\delta_j} d_{ij})/2\}$. Decrease d_{ij} and $d_{i'j'}$ by γ.

Step 2. (At this point (13) holds with equality.) Output the remaining pairs by using procedure BE-SPLIT on input values d'_{i1}, d'_{i2} defined as follows. If $\delta_i = 1$ then

$$d'_{j1} = d_{1j}, d'_{j2} = \sum_{i>1} d_{ij}, \text{ for } 1 \leq j \leq q.$$

If $\delta_j = 1$ then

$$d'_{i1} = d_{i1}, d'_{i2} = \sum_{j>1} d_{ij}, \text{ for } 1 \leq i \leq p.$$

In both cases the pairs of the form ii' output by BE-SPLIT correspond to pairs of the form $ij, i'j'$ in the obvious way.

We now show PE-SPLIT outputs a complete pairing. Consider Step 1. First we show the indices in Step 1a exist. For any index $i \neq \delta_i$,

$$\sum_{j \neq \delta_j} d_{ij} \leq \sum_j d_{ij} \leq \sum_{i=\delta_i \text{ or } j=\delta_j} d_{ij} < \sum_{i \neq \delta_i} \sum_{j \neq \delta_j} d_{ij}.$$

The second inequality follows by the relation defining δ_i and δ_j (which holds throughout the algorithm). The third inequality restates the fact that (13) holds with strict inequality.

The displayed relation implies that $\sum_{j \neq \delta_j} d_{ij}$ is nonzero for at least two indices $i \neq \delta_i$, say i, i'. These indices have corresponding indices j, j' that can be used in Step 1a unless there is an index $r \neq \delta_j$ with $d_{ir}, d_{i'r} > 0$. To handle this case note that $\sum_{i \neq \delta_i} d_{ij}$ is nonzero for at least two indices $j \neq \delta_j$, say j, j'. (The proof is analogous to the previous one for i, i'.) Without loss of generality $j \neq r$. We can choose $i'' \neq \delta_i$ so that $d_{i''j} > 0$, and without loss of generality $i'' \neq i$. Hence the indices i, i'', r, j can be used in Step 1a.

Next observe that Step 1 eventually halts. This follows because the two sides of (13) have the same parity (by (12)). Each pair output maintains this relation. So eventually (13) holds with equality and Step 1 halts.

To show Step 2 works correctly we need only check that the input conditions (10)–(11) for BE-SPLIT hold. (10) follows from equality in (13). Observe that (11) is equivalent to $d_{j1} + d_{j2} \leq (\sum_i d_{i1} + d_{i2})/2$ for every index j. If $\delta_i = 1$ then

$$d'_{j1} + d'_{j2} = \textstyle\sum_i d_{ij} \leq \sum_j d_{1j} = (\sum_{i,j} d_{ij})/2$$

which gives the equivalent form of (11). The same argument applies to the case $\delta_j = 1$.

Next we show that PE-SPLIT outputs a small number of pairs. Specifically only $O(p)$ pairs ii' have indices jj' for which $\gamma_{ij,i'j'}$ is positive.

To prove this first consider two indices i, i' chosen in Step 1a. Without loss of generality assume this is not the last execution of Step 1a. This implies that when Step 1b halts one of the following conditions holds:

(i) $d_{ij} = 0$ for every index $j > \delta_j$, or a similar condition holds for i';
(ii) there is an index j such that $d_{ij'} = d_{i'j'} = 0$ for every index $j' \neq j, j' > \delta_j$.

Before Step 1a neither condition holds. So condition (i) holds at most once for a given index i. For (ii) note that at least one of the indices i, i', say i, has more than one positive value d_{ij}, $j > \delta_j$, before Step 1a, but only one positive value after Step 1b. So condition (ii) holds at most once for a given index i. We conclude Step 1a is executed for $O(p)$ distinct pairs i, i'.

Next consider Step 2. If $\delta_i = 1$ then every pair $ij, i'j'$ output in Step 2 has $i = 1$. So there are $O(p)$ distinct pairs i, i' in Step 2. If $\delta_j = 1$ the input to BE-SPLIT consists of p values d'_{i1}, d'_{i2}. Thus BE-SPLIT outputs $O(p)$ distinct pairs ii'. Each such pair corresponds to pairs $i1, i'j$, $1 < j \leq q$ output by PE-SPLIT. So again Step 2 outputs $O(p)$ distinct pairs i, i'.

PE-SPLIT can be implemented to use total time $O(p + q + v)$, where v is the number of positive values d_{ij}. To implement Step 1 we maintain $q + 1$ lists corresponding to these sets:

$$T = \{i : i > \delta_i, \text{ at least two indices } j > \delta_j \text{ have } d_{ij} > 0\}.$$

$$O_j = \{i : i > \delta_i, j \text{ is the unique index } > \delta_j \text{ with } d_{ij} > 0\}, \delta_j < j \leq q.$$

In addition each index i has a list of the indices j that have $d_{ij} > 0$. If $|T| \geq 2$ Step 1a chooses two indices i, i' from T. The lists for i and i' are used to generate the indices j, j' for Step 1a. If $|T| \leq 1$ Step 1a chooses indices i and i' from two distinct lists (among T and the O_j) and generates indices j, j' in a similar way. The implementation of Step 2 is straightforward.

This concludes the discussion of PE-SPLIT (we have also proved the theorem on p, q-party graph degree sequences).

6.4.2 General targets

We adapt the algorithm PE-SPLIT to P-SPLIT, the routine for general partitions and arbitrary k. As before assume the given values d_{ij}, $i = 1, \ldots, p$, $j = 1, \ldots, q$ satisfy (12)–(13). Procedure P-SPLIT outputs a complete pairing, which satisfies additional constraints coming from the graph H associated with the input values. Recall that index i corresponds to vertex x_i, with $d_{ij} = d_j(s, x_i)$. The pair $ij, i'j'$ corresponds to split $sx_i, sx_{i'}$. P-SPLIT must limit the number of foolish splits corresponding to its pairs; the precise sense of this requirement is given in Lemmas 6.8–6.9 below.

Recall that the indices of splits $sx_i, sx_{i'}$ are chosen in Step 1a of PE-SPLIT and Step 1a of BE-SPLIT. The latter is executed when BE-SPLIT is called from Step 2 of PE-SPLIT. The P-SPLIT routine uses a rule, called the Choice Step, to specify how the indices of splits are chosen. Note that if BE-SPLIT is called with $\delta_i = 1$ in PE-SPLIT there is no need for such a rule, since in this case there is a unique complete splitting. So BE-SPLIT uses the Choice Step only when $\delta_j = 1$.

To use the same Choice Step for both Steps 1a we extend some notation from PE-SPLIT to BE-SPLIT. For BE-SPLIT define $\delta_i = 1$ and $\delta_j = 0$. Observe that these choices make the first sentence of Step 1a of BE-SPLIT identical to Step 1a of PE-SPLIT. (Although this convention allows us to treat the two routines uniformly, BE-SPLIT is not a special case of PE-SPLIT: the original definition of δ_i and δ_j for PE-SPLIT would lead us to choose $\delta_i = 0, \delta_j = 1$ because of (10).) Also define $q = 2$ for BE-SPLIT.

We need some more notation for the Choice Step. A *normal execution* of Step 1a is an execution of Step 1a of P-SPLIT or an execution of Step 1a of BE-SPLIT when BE-SPLIT is called with $\delta_j = 1$ in P-SPLIT. Observe that in a normal execution of Step 1a an index i in $1 \leq i \leq p$ corresponds to a vertex of H. (This is false for an abnormal execution.) For $\delta_j < j \leq q$ define sets

$$I_j = \{i : \delta_i < i \leq p,\ d_{ij} > 0\}.$$

Also for an index $\delta_i < i \leq p$ let F_i denote the set of all indices $f > \delta_i$ with sx_i, sx_f a foolish split in the associated graph H. (This makes sense in a normal execution.)

The routine P-SPLIT is the same as PE-SPLIT except that normal executions of Step 1a choose indices i, i' by executing the following Choice Step:

Choice Step. Execute the case below that applies.
Case 1: There are two distinct indices $j, j' > \delta_j$ with $I_j \cap I_{j'} \neq \emptyset$.
Choose any $i \in I_j \cap I_{j'}$. Choose $i' \in \cup_t I_t - i$, with $i' \notin F_i$ if possible.

Case 2a: There are exactly two nonempty sets I_j, disjoint and both of cardinality two.
Case 2b: There are exactly three nonempty sets I_j, pairwise disjoint and all of cardinality one.

In both Cases 2a–b, choose any index i in a set I_j such that $\cup_{t\neq j} I_t - F_i$ is nonempty, and choose an index i' in the latter set.

Case 3: No previous case applies.

Choose index $j > \delta_j$ so that I_j is nonempty and $|I_j|$ is minimum. Choose any $i \in I_j$. Choose an index $i' \in \cup_{t\neq j} I_t$, with $i' \notin F_i$ if possible.

When the target k is even there are no foolish splits, so the Choice Step does not substantively change PE-SPLIT. Also note that Case 2b never holds in BE-SPLIT. The analysis of PE-SPLIT shows P-SPLIT outputs a complete pairing if it is well-defined, i.e., the index i always exists in Case 2. The following two lemmas state properties of the pairs output by P-SPLIT in terms of the corresponding splits on H, the graph corresponding to the input for P-SPLIT. As in Section 6.3 H satisfies (1). Let H have vertex set $V_H + s$.

Lemma 6.8 *(a) P-SPLIT is well-defined.*
(b) P-SPLIT outputs at most $C = 3$ foolish splits.

Proof: Consider a normal execution of Step 1a. We discuss each case of the Choice Step.

Case 1: There are two distinct indices $j, j' > \delta_j$ with $I_j \cap I_{j'} \neq \emptyset$.

For any $i \in I_j \cap I_{j'}$, $d(s, x_i) \geq d_{ij} + d_{ij'} \geq 2$. Hence vertex x_i is in at most one foolish split and $|F_i| \leq 1$ (recall the discussion preceding the Choice Step in Section 6.3). Thus Case 1 does not create a foolish split when $|\cup_t I_t| \geq 3$. We conclude that Case 1 creates a foolish split only when $|\cup_t I_t| \leq 2$.

Case 2a: There are exactly two nonempty sets I_j, disjoint and both of cardinality two.
Case 2b: There are exactly three nonempty sets I_j, pairwise disjoint and all of cardinality one.

We must show some index i in a set I_j has $\cup_{t\neq j} I_t - F_i$ nonempty. Suppose on the contrary that for every $j > \delta_j$ and $i \in I_j$, $\cup_{t\neq j} I_t \subseteq F_i$. Thus $d(x_i) = k$ and $d(x_i, x_r) = h$ for every $r \in \cup_{t\neq j} I_t$. The hypotheses of Cases 2a and 2b both imply there are two such indices r. Thus $d(s, x_i) = 1$.

Let $X = V_H - \{x_i : i \in \cup_j I_j\}$. Observe that X is nonempty: If $\delta_i = 1$ then $x_1 \in X$. If $\delta_j = 1$ then there are vertices y with $d_1(s, y) > 0$. Such a y does not correspond to an index in any I_j since as noted above, any $i \in I_j$ has $d(s, x_i) = 1$, so $d(s, x_i) = d_j(s, x_i)$.

The equations of the first paragraph of this case imply $d(X, V_H - X) = 0$. We claim $d(s, X) \leq 2$. The claim implies $d(X) = d(s, X) \leq 2$. Using $k \geq 3$ this violates (1), giving the desired contradiction.

To prove the claim first suppose $\delta_i = 1$. In this case $d(s, X) = d(s, x_1)$. For Step 1a of P-SPLIT strict inequality holds in (13), so $d(s, x_1) \leq 4 - 2 = 2$. This inequality is also valid for Step 1a of BE-SPLIT since strict inequality holds in (11). Next suppose $\delta_j = 1$. In this case $d(s, X) = d_1(s, X)$. Again since strict inequality holds in (13), $d_1(s, X) \leq d_1(s, V_H) \leq 4 - 2 = 2$.

In summary we have proved the claim, thus showing the desired index i of the Choice Step exists. This gives part (a) of the lemma. We have also shown this case does not create a foolish split.

Case 3: No previous case applies.

Suppose $|\cup_j I_j| \geq 4$. We observe that selecting j as in the Choice Step gives $|\cup_{s\neq j} I_s| \geq 3$. In proof first note the sets I_j are pairwise disjoint, by Case 1. The observation follows if $|I_j| \geq 3$, since there is at least one set I_t. If $|I_j| = 2$ the observation follows from the definition of Case 2a. If $|I_j| = 1$ the observation follows from the supposition $|\cup_j I_j| \geq 4$.

The observation implies that any $i \in I_j$ has an index $i' \in \cup_{t\neq j} I_t$ not creating a foolish split. We conclude that Case 3 creates a foolish split only when $|\cup_j I_j| \leq 3$.

We have shown the Choice Step does not create a foolish split unless $|\cup_j I_j| \leq 3$. This implies Step 1a of P-SPLIT creates at most one foolish split (recall the discussion preceding the Choice Step in Section 6.3). If $\delta_i = 1$ in P-SPLIT then BE-SPLIT creates at most one foolish split (since every split involves vertex 1). Suppose $\delta_j = 1$ in P-SPLIT. Then as above, Step 1a of BE-SPLIT creates at most one foolish split. In Step 2 of BE-SPLIT all the splits output involve a common vertex (the vertex indexed as 1 in BE-SPLIT). Thus Step 2 of BE-SPLIT creates at most one foolish split. In summary at most three foolish splits are created, giving part (b) of the lemma. $\square$

For the next lemma assume G and H satisfy the invariant of Section 6.3 (we show this is the case in the next section).

Lemma 6.9 *If the first split output by P-SPLIT is foolish then executing a maximal admissible subset of the splits output gives a new graph G containing a C_4-obstacle with $d(s) = 4$.*

Proof: The main case is when the first split is output in a normal execution of Step 1a. We start by showing that if this is not the case, i.e., the first split is output in Step 1a of an abnormal execution of BE-SPLIT or Step 2 of BE-SPLIT, then the first split is not foolish. In fact we show BE-SPLIT finds a complete admissible splitting. To prove this observe that in both of these situations

$$d(s, x_1) = \sum_{i>1} d(s, x_i).$$

This follows because if the first split is output in BE-SPLIT with $\delta_i = 1$ in P-SPLIT, equality holds in (13). Similarly if the first split is output in Step 2 of BE-SPLIT with $\delta_j = 1$ in P-SPLIT, equality holds in (11). (In this case x_1 is the vertex indexed 1 in BE-SPLIT.) The displayed equation implies H has a unique complete splitting with no split joining a vertex to itself. Hence this splitting is found by BE-SPLIT. The splitting is admissible by the second remark at the end of Section 6.1.

Now we turn to the main case, when the first split is output in a normal execution of Step 1a. Fix the sets I_j to their values right before this first split is output. We claim the first split is foolish only when exactly two of these sets are nonempty, say I_j and $I_{j'}$, and for distinct indices $a, b, c > \delta_i$,

$$I_j = \{a\} \text{ and } I_{j'} = \{b, c\}.$$

Cases 1–3 of Lemma 6.8 show the claim holds if $|\cup_j I_j| \geq 4$, or if $|\cup_j I_j| = 3$ and either two sets I_j have a nonempty intersection or the nonempty sets I_j are pairwise disjoint singletons. Since the indices of Step 1a exist, $|\cup_j I_j| \geq 2$. Hence the remaining case to prove the claim is $|\cup_j I_j| = 2$. Let $\cup_j I_j = \{i, i'\}$. We must show that the split $sx_i, sx_{i'}$ is not foolish. We will show it is admissible.

First observe that because the execution of Step 1a is normal, $d(s, \{x_i, x_{i'}\}) > d(s)/2$. (If $\delta_i = 1$ then $d(s, \{x_i, x_{i'}\})$ equals the right-hand side of (13) or (11). If $\delta_j = 1$ then $d(s, \{x_i, x_{i'}\})$ is at least the right-hand side of (13).) This implies any complete splitting with no split joining a vertex to itself contains the split $sx_i, sx_{i'}$. Hence this split is admissible.

We have proved the claim. To prove the lemma it remains only to treat the case of sets $I_j, I_{j'}$ displayed above.

Suppose for these sets, the first split output by P-SPLIT is foolish. The Choice Step takes $i = a$, so $I_{j'} \subseteq F_a$. This implies $d(x_a) = d(x_b) = d(x_c) = k$, $d(x_a, x_b) = d(x_a, x_c) = h$ and $d(s, x_a) = 1$.

The first split is output in an execution of BE-SPLIT. This follows because strict inequality in (13) is impossible as it implies this contradiction:

$$d_{j'}(s, \{x_b, x_c\}) \leq d_{j'}(s, V_H) \leq d_j(s, x_a) + d_{j'}(s, \{x_b, x_c\}) - 2 = d_{j'}(s, \{x_b, x_c\}) - 1.$$

We conclude that the split is output in BE-SPLIT, so $\delta_i = 1$ and (11) holds with strict inequality.

Define $A = V_H - \{x_a, x_b, x_c\}$. This set is nonempty since it contains x_1.

The criticality of x_b implies

$$d(x_b, A) \leq h + 1 - d(s, x_b).$$

Similarly $d(x_c, A) \leq h + 1 - d(s, x_c)$ and $d(x_a, A) = 0$. Since strict inequality holds in (11),

$$d(s, x_1) \leq d(s, x_b) + d(s, x_c) + d(s, x_a) - 2 = d(s, x_b) + d(s, x_c) - 1.$$

These inequalities imply $d(A) = d(A, x_b) + d(A, x_c) + d(A, x_a) + d(s, x_1) \leq 2h + 1 = k$. Hence by (1), equality holds throughout. Equality in the first displayed inequality implies $d(x_b, x_c) = 0$. Equality in the second displayed inequality implies $d(s, x_1) + d(s, x_a) = d(s, x_b) + d(s, x_c) = d(s)/2$. By (10), $d(s, x_1) = d_j(s, x_1)$. Examining the definition of d'_{ij} (for $\delta_j = 1$) in Step 2 of P-SPLIT shows that partition 1 in BE-SPLIT is the same as the original partition 1 in H. Let X_1, X_a, X_b, X_c denote the set of all vertices of G contained in x_1, x_a, x_b, x_c respectively. In graph G either $d(s, X_1 \cup X_a) = d_1(s, X_1 \cup X_a)$ or $d(s, X_b \cup X_c) = d_1(s, X_b \cup X_c)$. Now we have deduced that A, X_a, X_b, X_c is a C_4-obstacle in G.

Without loss of generality, Step 1a outputs the (foolish) split sx_a, sx_b with multiplicity one. This gives equality in (11). The remaining splits are sx_1, sx_c with multiplicity $d(s, x_c)$ and sx_1, sx_b with multiplicity $d(s, x_b) - 1$. It is easy to see that a maximal admissible splitting executes all these splits except the split sx_a, sx_b and one copy of the split sx_1, sx_c (use Proposition 6.1). This gives the lemma. □

Since P-SPLIT is a special case of PE-SPLIT, it has the same bound on the number of pairs output, specifically, $O(p)$ pairs ii' have indices jj' for which $\gamma_{ij,i'j'}$ is positive. It is easy to implement P-SPLIT in time $O(m + n)$. We use the same data structures as for BE-SPLIT.

6.5 Computing a near-complete allowed splitting

Lemma 5.1 shows that any graph has a splitting of allowed pairs that is either complete or gives a graph containing a C_4-obstacle with $d(s) = 4$. We call this a *near-complete allowed splitting*. This section adapts the algorithm of Section 6.3 to find such a splitting.

We note that Nagamochi and Ibaraki [17] give an algorithm for this task when k is even. The algorithm achieves the same time bound as ours in the special case that all edges have unit capacity.

The new algorithm is a straightforward extension of GENERAL-SPLIT, but we state it below for convenience. Recall the notation introduced at the start of Section 6.4 to represent partitions in contracted graphs: $d_i(s, x)$ for degrees and sx_i, sy_j for splits.

The procedure TRANSLATE is called with two disjoint sets $X, Y \subset V$ and integers i, j, γ. X and Y have been contracted to vertices in H, and the split sX_i, sY_j with multiplicity γ is admissible in H.

Procedure TRANSLATE(X, i, Y, j, γ)
Repeat Step 1 until $\gamma = 0$.

Step 1. Choose a vertex x of G corresponding to X: If X is a vertex of G let $x = X$. Otherwise choose a vertex $x \in X$ that is a neighbour of s in partition class P_i. Similarly choose a vertex y of G corresponding to Y, in partition class P_j. Execute split sx, sy in G with multiplicity $\gamma_0 = \min\{d(s, x), d(s, y), \gamma\}$ (d is the degree function in G). Decrease γ by γ_0.

Procedure PARTITION-SPLIT is our allowed splitting algorithm. It executes a near-complete allowed splitting at s on the original graph G. It starts with H initialized to the given graph G.

Procedure PARTITION-SPLIT
Repeat Steps 1–3 until either $d(s) = 0$ or $d(s) = 4$ and H consists of four vertices forming a C_4-obstacle in G.

Step 1. Use graph G to compute the degree values $d_i(s, x)$ for each vertex x of H. Call P-SPLIT (with these degree values $d_i(s, x)$) to find a complete splitting at s in H, with splits sx_i, sy_j.

Step 2. Call HOOK-UP to execute a maximal admissible subset of these splits in H, and to return the corresponding family of dangerous sets $\mathcal{Y}$. For each split that was executed in H call TRANSLATE to execute it in G.

Step 3. In H do the following for each set $Y \in \mathcal{Y}$: If Y is critical, contract Y. Otherwise ($d(Y) = k + 1$) test if Y has a k-bisection. If it does not, contract Y. If it does, say into sets A and B, contract A and contract B.

To show correctness first observe that PARTITION-SPLIT maintains the same invariant as GENERAL-SPLIT, by exactly the same argument. Also observe that P-SPLIT is called with degree values $d_i(s, x)$ that satisfy (12)–(13): (12) holds since $d(s)$ is always even. (13) holds in the first iteration of PARTITION-SPLIT because of the initial choice of edges sx. (This is done by the algorithm of Lemma 4.1, which we implement in Section 6.6.) Now we show (13) holds in subsequent iterations. For $\delta_i = 1$ (13) holds because every neighbour of s is in a dangerous set. Suppose $\delta_j = 1$. It suffices to show that each iteration preserves the fact that any partition j has $\sum_i d_{ij} \le d(s)/2$. This holds because each split involves two distinct partition classes, so each unexecuted split increases $\sum_i d_{ij}$ for two different partitions j.

It remains only to show that the main loop terminates the algorithm correctly, which we now do. Call an iteration of PARTITION-SPLIT *normal* if it performs at least one normal execution of Step 1a, i.e., P-SPLIT starts with either strict inequality in (13) or equality in (13), $\delta_j = 1$ and strict inequality in (11). We will show that an execution of PARTITION-SPLIT begins with a finite number of normal iterations starting with inequality in (13), followed by a finite number of normal iterations starting with equality in (13) and inequality in (11), followed by at most one abnormal iteration.

First observe that once (13) holds with equality, it holds with equality throughout the rest of the algorithm. This follows since equality means $d(s, x_1) = \sum_{i>1} d(s, x_i)$ or $d_1(s, V) = \sum_{j>1} d_j(s, V)$, and both of these relations are preserved by all subsequent splits and contractions.

Next observe that an iteration starting with equality in both (13) and (11) is abnormal. Furthermore an abnormal iteration is the last iteration of PARTITION-SPLIT. This was shown in the first paragraph of the proof of Lemma 6.9.

It remains only to show that there are a finite number of normal iterations. The finiteness property holds because of a property similar to GENERAL-SPLIT: For indices $j \ge 0$ define

$$N_j = \{x : \text{vertex } x \text{ of } H \text{ has } d_i(s, x) > 0 \text{ for at least } j \text{ partition classes } i\}.$$

Thus N_0 is V_H and N_1 is the set of neighbours of s in H. Also $N_j = \emptyset$ for $j \ge 3$ in BE-SPLIT, i.e., when (13) holds with equality. The property is that with two exceptions, every normal iteration ending with strict inequality in (13) or (11) decreases $|N_j|$ for some j in $0 \le j \le 3$. The two exceptions are the last iteration, and an iteration that starts with strict inequality in (13) and ends with equality in (13). Note that the sets N_j in this termination property are defined using the given partition in iterations that start with inequality in (13), and the partition for BE-SPLIT in iterations that start with equality in (13). The exceptional iteration just mentioned switches the definition of sets N_j, since subsequent iterations execute BE-SPLIT. Note that this switch does not increase any $|N_j|$, since it just merges partition classes.

Our termination property implies there are a finite number of normal iterations: Once a given vertex leaves N_j, $j \ge 1$, it never reenters N_j. Hence a sequence of normal iterations that preserves the number of vertices of H has finite length and leads to a decrease of $|N_0|$. Now we prove the termination property. Let the first split output by P-SPLIT be sx, sy with multiplicity γ. We begin by assuming this split is not foolish.

Suppose HOOK-UP splits off γ pairs of edges sx, sy. If sx, sy is the last split output in Step 1 of P-SPLIT or BE-SPLIT then after executing this split equality holds in (13) or (11) respectively. So assume sx, sy is not the last split output in Step 1. Assume N_0 does not change in this iteration (else we are done). In Step 1 assume vertices x and y correspond to indices i and i' respectively. Recall the conditions (i) and (ii) that hold when Step 1b of P-SPLIT halts (see the end of Section 6.4.1). These imply that (without loss of generality) one of these conditions holds when Step 1b of P-SPLIT halts: (i) $d_j(s, x) = 0$ for every index $j > \delta_j$; (ii) $d_j(s, x) > 0$ for more than one index $j > \delta_j$ before Step 1a but for only one such index after Step 1b. It is easy to see that one of these conditions also holds after Step 1a of BE-SPLIT. We conclude that the iteration decreases $|N_r|$ for some index r, $1 \le r \le 3$: If (i) holds then $r = 1$ if $\delta_j = 0$ and $r = 1$ or 2 if $\delta_j = 1$. If (ii) holds then $r = 2$ if $\delta_j = 0$ and $r = 2$ or 3 if $\delta_j = 1$. (The assumption that N_0 does not change guarantees that at the end of the iteration x is still a vertex of H.)

On the other hand suppose HOOK-UP splits off fewer than γ pairs of edges sx, sy. HOOK-UP returns a set $Y \in \mathcal{Y}$ containing both x and y, and $Y \neq \{x, y\}$ since the split is not foolish. Hence the Contraction Step decreases $|N_0|$ when it processes set Y.

Now we consider the case that the first split sx, sy is foolish. We will show this is the last iteration of PARTITION-SPLIT. Lemma 6.9 implies that after HOOK-UP, graph G has $d(s) = 4$ and contains a C_4-obstacle. Let the obstacle consist of sets A_1, A_2, B_1, B_2, with a_1, a_2, b_1, b_2 their neighbours of s.

Without loss of generality the two splits that are not executed by HOOK-UP are sa_i, sb_i for $i = 1, 2$. HOOK-UP returns a dangerous set D_i containing a_i and b_i, for $i = 1, 2$. These sets are distinct, by Lemma 2.8, so they are disjoint.

Note the following general fact: If X is a critical set and Y is dangerous, with $X \cap Y$ containing a neighbour of s, then $X \subseteq Y$ or $Y \subseteq X$. (In proof suppose the contrary, i.e., $Y - X$ and $X - Y$ are nonempty. Then $k + (k + 1) \ge d(X) + d(Y) \ge d(X - Y) + d(Y - X) + 2d(s, X \cap Y) \ge k + k + 2$, a contradiction.)

The fact implies $A_i \cup B_i \subseteq D_i$ for $i = 1, 2$. Disjointness of the D_i implies $D_i = A_i \cup B_i$. This implies $d(D_i) = k + 1$ (Lemma 2.4). Thus A_i and B_i form the unique k-bisection of D_i. The Contraction Step makes each of the four obstacle sets A_i, B_i, $i = 1, 2$ into a vertex. Hence the termination condition of the main loop now detects a C_4-obstacle on four vertices. We have thus shown the termination property for PARTITION-SPLIT.

The efficiency analysis requires a stronger termination property, which we now prove: Define

$$\Phi = 2|N_1| + |N_2| + |N_3|.$$

The property states that Φ never increases, and every normal iteration except possibly two either decreases Φ or increases the number of foolish splits that exist in H. The two exceptional iterations are the last iteration, and an iteration that starts with strict inequality in (13) and ends with equality in (13).

To prove this property we analyze how Φ changes by computing first the change in Step 2, and then the change in Step 3. (First recall that Φ does not increase when we switch definitions of the N_j.) Step 2 can remove a vertex from a set N_j when it executes a split. Thus Step 2 can decrease Φ but it never increases Φ. To analyze Step 3 consider the effect of a contraction on Φ: Let vertices x_1 and x_2 be merged to form vertex X. Let $d_i(s, x_r) > 0$ for exactly j_r partition classes i, $r = 1, 2$, with $j_1 \ge j_2$. Then $d_i(s, X) > 0$ for between j_1 and $j_1 + j_2$ partition classes. Hence this operation does not change Φ if $j_2 = 0$ and it decreases Φ if $j_2 > 0$. We have now shown that an iteration never increases Φ.

The proof of the first termination property shows that with three exceptions the first split causes Φ to decrease, by either decreasing some $|N_j|$, $1 \le j \le 3$ or contracting a set that contains at least two neighbours of s. The three exceptions are the two exceptional iterations mentioned above, and when the first split creates a dangerous set with a k-bisection, whose two contractions do not change Φ. Now consider a normal iteration that is not one of the two exceptional iterations and does not change Φ. As in the proof of Lemma 6.6, every split that is foolish at the start of the iteration is foolish at the end of the iteration, and furthermore the first split causes a contraction creating a new foolish split. This proves the stonger termination property.

The efficiency analysis for PARTITION-SPLIT is essentially identical to previous arguments. PARTITION-SPLIT has $O(\log n)$ iterations, by the same derivation as for GENERAL-SPLIT: Lemma 6.5 is proved exactly the same way. Similarly for inequality (9). The proof of Lemma 6.6 incorporates the stronger termination property just proved. (This property implies that $s_i \leq C'$ in $O(1)$ iterations, since $s_i \leq C'$ implies $\Phi \leq 4C'$.)

The total number of distinct splits executed in Step 2 in all iterations is $O(n)$, for both G and H. The proof is the same argument as GENERAL-SPLIT. For the analysis of graph H we use the fact that each execution of P-SPLIT gives a splitting with $O(s_i)$ distinct splits sx, sy.

The time bound for PARTITION-SPLIT is $O(n(m+n\log n)\log n)$. (The time is $O(nm\log n)$ on unit capacity graphs.) The argument is the same as GENERAL-SPLIT, using the above facts.

6.6 Finding good neighbours

This section gives algorithms to compute the set of edges incident to s, i.e., the values $d(s,x)$. This problem occurs three times in our connectivity augmentation algorithm – in computing the initial graph, rearranging edges in a C_4-obstacle and rearranging edges in a C_6-obstacle. We discuss each of these. Our approach is to simulate the procedures given in previous sections using an efficient algorithm for computing the initial edges of an admissible splitting.

Nagamochi et. al. [18] give an algorithm MINIMAL that starts with a graph $G = (V+s, E)$ satisfying (1), and deletes edges incident to s so (1) still holds but decreasing any value $d(s,x)$ destroys (1). MINIMAL is a modification of HOOK-UP and has the same time bound, $O(n(m+n\log n))$. The time is $O(nm)$ on unit capacity graphs.

We use a straightforward generalization of MINIMAL. In addition to G, this algorithm is given a value $\ell(s,x)$ for each $x \in V$, with $\ell(s,x) \leq d(s,x)$ in G. The graph that is returned satisfies (1), has $\ell(s,x) \leq d(s,x)$ for every $x \in V$, and no value $d(s,x)$ can be decreased subject to these constraints. It is simple to modify the algorithm of [18] to handle these lower bounds. (Specifically the algorithm of [18] initializes the solution graph to G with each value $d(s,x)$ decreased to 0. It proceeds to add back a minimal set of edges of G incident to s that achieves (1). The only change for the modified algorithm is to initially decrease each $d(s,x)$ to $\ell(s,x)$.) The time bound for the modified algorithm is unchanged. We still call this modified algorithm MINIMAL.

6.6.1 Computing the initial graph

The initial edges incident to s are computed by Steps 1–2 of the procedure given in Lemma 4.1. (These steps are used for both even and odd k.) Step 1 computes values $d(s,x)$ that achieve (1) and are minimal. We do this using the original algorithm MINIMAL of [18]. The rest of Step 1 ensures that $d(s)$ is even, specifically $d(s) = 2\alpha$ for α as defined in Section 4. This part is trivial to implement.

Step 2 applies if there is a partition class P_1 such that $d(s,P_1) > d(s, V-P_1)$. It decreases $d(s,P_1)$ and increases $d(s,V-P_1)$ by the same amount, preserving (1), until either the two quantities become equal or any further change destroys (1). The first case ends with $d(s, P_1) = d(s, V - P_1) = \alpha$; the second case ends with $d(s, P_1) > \alpha > d(s, V - P_1)$. Our implementation of Step 2 begins by deciding which of these two alternatives holds, as follows.

Call the degree function at the end of Step 1 d_1. So $d_1(s, P_1) > \alpha > d_1(s, V - P_1)$. Define values $d_2(s, x)$ and lower bounds $\ell(s, x)$ by

$$d_2(s, x) = d_1(s, x), \ell(s, x) = 0 \text{ for } x \in P_1;$$

$$d_2(s, x) = \ell(s, x) = k \text{ for } x \in V - P_1.$$

Execute MINIMAL starting with these values. Let d_2' denote the degree function in the graph that is returned. Note that d_2' corresponds to a minimal degree function satisfying (1) for sets $X \subseteq P_1$.

Clearly when the first case of Step 2 holds, we get $d_2'(s, P_1) \le \alpha$. It is not hard to see that when the second case of Step 2 holds we get $d_2'(s, P_1) > \alpha$. (Use the family of sets W_i defined in the discussion of Step 2c.) We process the two cases as follows.

Case 1: $d_2'(s, P_1) \le \alpha$.

As in Step 2 our goal is to construct a new degree function so $d(s, P_1) = d(s, V - P_1) = \alpha$. To do this first define values $d_2''(s, x)$ for $x \in P_1$ as follows. Initialize each $d_2''(s, x)$ to $d_2'(s, x)$. Then increase some values $d_2''(s, x)$, $x \in P_1$, maintaining $d_2''(s, x) \le d_1(s, x)$, until $d_2''(s, P_1) = \alpha$. Note that aside from these constraints, the vertices x whose values are increased and the amounts increased are arbitrary. It is possible to achieve the desired function d_2'' by the definition of d_2 and Case 1.

Now define values $d_3(s, x)$ and lower bounds $\ell(s, x)$ by

$$d_3(s, x) = \ell(s, x) = d_2''(s, x) \text{ for } x \in P_1;$$

$$d_3(s, x) = k, \ \ell(s, x) = 0 \text{ for } x \in V - P_1.$$

Execute MINIMAL starting with these values. Use the resulting graph as the output of Step 2.

To see this construction is correct, observe that any $x \in P_1$ has $d_1(s, x) \ge d_2''(s, x)$. Suppose we execute Step 2 starting with the degree function d_1. In each execution of Step 2b we can choose u to be any vertex x satisfying $d_1(s, x) > d_2''(s, x)$, where the quantity on the left is the current value of d_1. This follows from our characterization of d_2' as a minimal degree function satisfying (1) for sets $X \subseteq P_1$. Hence we can execute Step 2 so it halts with a degree function d equal to d_2'' on vertices of P_1. This degree function has $d(s) = 2\alpha$. The existence of d implies that the above execution of MINIMAL returns a similar degree function d_3', i.e., d_3' equals d_2'' on P_1 and $d_3'(s) = 2\alpha$. Thus our algorithm behaves the same as Step 2.

Case 2: $d_2'(s, P_1) > \alpha$.

Our characterization of d_2' shows that Step 2 halts with a degree function satisfying $d(s, P_1) = d_2'(s, P_1) = d(s, V - P_1)$. So it suffices to construct a degree function satisfying this equation (and (1)). To do this define values $d_3(s, x)$ by

$$d_3(s, x) = d_2'(s, x) \text{ for } x \in P_1;$$

$$d_3(s,x) = k \text{ for } x \in V - P_1.$$

Execute MINIMAL (with no lower bounds) starting with these values. Let d_3' be the resulting degree function. Note that $d_3'(s, V - P_1) < d_3'(s, P_1)$ (by Step 2). Now as in Step 2c, increase values $d_3'(s,x)$ for $x \in V - P_1$ to make $d_3'(s, V - P_1) = d_3'(s, P_1)$.

We have shown this implementation of Step 2 is correct. It constructs the initial degree function in the same time bound as MINIMAL, $O(n(m + n \log n))$.

6.6.2 Rearranging obstacles

The basic step in rearranging obstacles operates on a given critical set $X \subset V$ and a given class P_i such that all edges from s to X go to vertices of P_i. The task is to check if this is forced, more precisely, if X has a subpartition $\mathcal{F}$ consisting of sets U, each contained in P_i, such that in $G - s$,

$$k - d(X) = \sum_{U \in \mathcal{F}} k - d(U). \tag{14}$$

If no such $\mathcal{F}$ exists, we may execute a second part of this basic step: The second part replaces one edge that goes from s to $X \cap P_i$ by an edge from s to $X \cap P_j$, $j \neq i$, still preserving (1).

We perform the basic step by computing a minimal degree function satisfying (1) for all subsets of $X \cap P_i$. Specifically, define values $d_1(s,x)$ and lower bounds $\ell(s,x)$ by

$$d_1(s,x) = d(s,x),\ \ell(s,x) = 0 \text{ for } x \in X \cap P_i;$$

$$d_1(s,x) = \ell(s,x) = k \text{ for } x \in V - (X \cap P_i).$$

Execute MINIMAL starting with these values. The resulting degree function d' is a minimal degree function satisfying (1) for subsets of $X \cap P_i$. By [6], $d'(s, X \cap P_i)$ equals the maximum possible value of the sum in (14). Note that the left-hand side of (14) (which is computed in $G - s$) equals $d(s,X)$ (since X is critical). We conclude that the condition of (14) is equivalent to the test $d(s,X) = d'(s, X \cap P_i)$.

If the test of (14) fails then $d(s,X) > d'(s, X \cap P_i)$. In this case the second part of the basic step, replacing an edge from s to X, can be performed as follows. Choose any vertex $u \in X \cap P_i$ with $d'(s,u) < d(s,u)$. Consider an irreducible subpartition of X (as defined before Lemma 5.7) that satisfies (14) (here we do not restrict the sets U to be contained in P_i). Vertex u is in some set U of the partition. We cannot have $U \subseteq P_i$ since then $d'(U) < d(U) = k$. Choose any vertex $v \in U - P_i$. Lemma 2.12 shows that replacing edge su by sv preserves (1).

To perform the procedure just described, use the above vertex u to define values δ_x for $x \in X \cap P_i$ by $\delta_x = 1$ for $x = u$ and $\delta_x = 0$ otherwise. Define values $d_2(s,x)$ and lower bounds $\ell(s,x)$ by

$$d_2(s,x) = \ell(s,x) = d(s,x) - \delta_x \text{ for } x \in X \cap P_i;$$

$$d_2(s,x) = d(s,x) + 1,\ \ell(s,x) = d(s,x) \text{ for } x \in V - (X \cap P_i).$$

Execute MINIMAL starting with these values. The resulting degree function replaces edge su by an edge sv, preserving (1), as described above.

The above basic step is used to rearrange obstacles as follows. First consider a C_4-obstacle in $G = (V + s, E)$. Let the obstacle sets be A_1, A_2, B_1, B_2, where partition class P_1 contains all neighbours of s in $A_1 \cup A_2$. According to Lemma 5.7 either the obstacle sets form a C_4-configuration in the given graph, or rearranging up to two edges incident to s gives a graph with no C_4-obstacle. We implement the procedure of Lemma 5.7 as follows. Use the basic step to check if both sets A_1 and A_2 have the subpartition $\mathcal{F}$ of (14). If so, the obstacle sets form a C_4-configuration. If not, perform the second part of the basic step (to replace one edge) on one set not having such a subpartition. Repeat this procedure for sets B_1 and B_2 if some partition class P_2 contains all neighbours of s in $B_1 \cup B_2$.

A C_6-obstacle is handled similarly. We implement the procedure of Lemma 5.6: Use the basic step to check if each of the six obstacle sets has the subpartition $\mathcal{F}$ of (14) (this subpartition will contain just one set). If so, the obstacle sets form a C_6-configuration. If not, perform the second part of the basic step on one obstacle set not having such a subpartition.

6.7 The constrained connectivity augmentation algorithm

Now we combine the pieces of the previous sections to get our algorithm for partition-constrained edge-connectivity augmentation.

The algorithm uses a subroutine to find a complete allowed splitting in a graph having $d(s) = 6$, $d(s, P) \leq 3$ for every partition class P, and containing no obstacle. The splitting exists by Theorem 3.6. To find it try every possible complete splitting that respects the partition constraints. Test every resulting graph for k-edge-connectivity using the procedure of [16]. Return the desired complete allowed splitting.

The algorithm for partition-constrained edge-connectivity augmentation starts by executing the algorithm of Section 6.6 to define the initial degree function $d(s, x)$.

Next execute PARTITION-SPLIT. If it finds a complete allowed splitting then return. (This will be the case when the target connectivity k is even.) If the splitting is not complete then PARTITION-SPLIT finds a C_4-obstacle with $d(s) = 4$. As in Theorem 5.8 consider the two cases.

Case 1: Every split that was executed joins vertices in different obstacle sets.

Test if the C_4-obstacle gives a C_4-configuration, using the procedure of Section 6.6. If so, use the procedure of Lemma 5.1 to add three edges to G (the graph returned by PARTITION-SPLIT) to make it k-edge-connected. This is optimal, so return.

If there is no C_4-configuration, rearrange the C_4-obstacle using the procedure of Section 6.6. Execute PARTITION-SPLIT on the new graph. If it finds a complete splitting then return. If not, PARTITION-SPLIT finds a C_4-obstacle to which Case 2 applies, so execute that case.

Case 2: A split joining two vertices in the same obstacle set was executed.

Let sx, sy be such a split. Undo it to get a graph G' with $d(s) = 6$. If neither x nor y belongs to a partition P_i with $d(s, P_i)$ maximum in G' then G' contains has no obstacle. Find a complete allowed splitting using the subroutine described above.

In the opposite case change the edges of this split, as described in Theorem 3.6 Case 1, and execute the new split. This gives a graph with $d(s) = 4$. Execute PARTITION-SPLIT on this graph. If PARTITION-SPLIT finds a complete splitting then return. (This will be the case when there are only two partition classes.) If not, PARTITION-SPLIT finds a second C_4-obstacle. Use the two C_4-obstacles to construct a C_6-obstacle, as described in Lemma 3.9.

Suppose the C_6-obstacle is not in the original graph. Change a preceding split to get a graph G'' with no obstacle and $d(s) = 6$, as in the proof of Lemma 3.5. Find a complete allowed splitting in G'' using the subroutine.

Suppose the C_6-obstacle is in the original graph. Test if the C_6-obstacle gives a C_6-configuration, using the procedure of Section 6.6. If not, rearrange the C_6-obstacle using the procedure of Section 6.6. This gives a graph G'' with no obstacle and $d(s) = 6$. Find a complete allowed splitting of G'' by using the subroutine.

The remaining case is when the original graph has a C_6-configuration. Using the last C_4-obstacle found, execute the procedure of Lemma 5.1 to add three more edges to make the graph k-edge-connected. This is optimal, so return.

In summary we have proved the following.

Theorem 6.10 *The edge-connectivity augmentation problem with partition constraints can be solved for $k \geq 2$ in time $O(n(m + n \log n) \log n)$ and space $O(m)$. The time is $O(nm \log n)$ on unit capacity graphs.* □

7 An application in statics

In this section we show how one of our results (Corollary 5.10) solves an open question raised by A. Frank (and implicitly by A. Recski in his book [20, pages 63-64]) and hence present a possible application in statics, a new field where connectivity augmentation algorithms may be useful. Corollary 5.10 gives a good characterization for the augmentation problem where the starting graph G is bipartite and the goal is to increase the edge-connectivity to k optimally by adding edges such that bipartiteness is preserved. (The polynomial algorithm which solves this problem is a simplified version of our main algorithm, see Theorem 6.10.)

One of the basic 2-dimensional structures in statics is a *square-grid framework* consisting of horizontal and vertical *rods*, each of the same length and rigid, and *joints*, which connect the incident rods and which are rotatable. The rods collectively join all gridpoints in a rectangular region that are adjacent horizontally or vertically.

A square-grid framework can be deformed by rotating certain parts along certain joints. To prevent these deformations we can add rods (call them *extra rods*) to the framework diagonally into some of the squares. A framework (possibly containing extra rods) is *rigid* if, roughly speaking, it has no nontrivial deformations, that is, fixing the position of a rod of the grid to the plane, the positions of all other rods (and joints) are uniquely determined. (For a precise description and more details see [20, Section 2.6].)

How can we decide whether or not a framework (with extra rods) is rigid? The answer – which was found by Bolker and Crapo – depends only on the *graph of the framework*. Given a square-grid framework S with m rows and l columns of squares, define the graph $G_S = (V, E)$ of S as follows. Let $V = \{v_1, ..., v_m, w_1, ..., w_l\}$ and connect v_i and w_j by an edge if there is an extra rod in the intersection of row i and column j. Observe that G_S is a bipartite graph and if at most one extra rod may be put into each square, G_S is simple as well.

Theorem 7.1 *[2] The square-grid framework S is rigid if and only if the corresponding graph G_S is connected.* □

One may want to brace a framework, by adding more extra rods, so it can survive the failure of $k' \geq 1$ extra rods, that is, removing any set of at most k' extra rods, the framework remains rigid. We say that a square-grid framework S is *k-rigid* if S remains rigid by removing any set of at most $k' < k$ extra rods. This yields the following optimization problem: Given a square-grid framework S, find a smallest set F of new extra rods whose addition makes S k-rigid. In terms of graphs this corresponds to the augmentation problem we described at the beginning of this section and solved (without the additional simplicity assumption) in Corollary 5.10 and Theorem 6.10.

In the rest of this section we briefly discuss the case where simplicity must also be preserved. We show that if $k \leq 3$ then the (size of the) solution we obtain by our algorithm of Section 6 is optimal in this case, too, by showing how the possible parallel edges in our solution can be substituted one by one by some other new edges maintaining k-edge-connectivity and achieving simplicity.

Let $G = (A, B, E)$ be a simple bipartite graph we want to make optimally k-edge-connected preserving bipartiteness and simplicity. We can assume that $|A|, |B| \geq k$ and $k \geq 2$. Let F be a solution produced by our algorithm of Section 6 and suppose that $G' = (A, B, E \cup F)$ contains parallel edges.

First let $k = 2$ and let $f \in F$ be an edge which is parallel to some edge $e \in E \cup F$.

Lemma 7.2 *There exists and edge $f' \notin E \cup F$ for which $G'' = (A, B, E \cup F - f + f')$ is bipartite and 2-edge-connected.*

Proof: By the optimality of F the edge f is critical wrt. 2-edge-connectivity in G'. Since e is parallel to f we can observe that the only cut-edge in $G' - f$ is e. Let X and Y be the two connected components we obtain from $G' - f$ by removing e. It is enough to show that there exist two vertices u, v, one in X and one in Y which are not adjacent in $G' - f$ and which belong to different classes of the bipartition (A, B). The existence of such a vertex pair can be checked easily (using $|A|, |B| \geq 2$). □

Consider the case $k = 3$. Using the same notation suppose that G' contains parallel edges and let $f \in F$ be an edge which is parallel to some edge $e \in E \cup F$.

Lemma 7.3 *There exists and edge $f' \notin E \cup F$ for which $G'' = (A, B, E \cup F - f + f')$ is bipartite and 3-edge-connected.*

Proof: Let $f = xy$ and let $X \subset A \cup B$ ($Y \subset A \cup B$) be a minimal set containing x (y) with $d_{G'-f}(X) = 2$ ($d_{G'-f}(Y) = 2$, respectively). Minimality implies $X \cap Y = \emptyset$. The key observation is that $X \cup Y = A \cup B$ since $d_{G'}(X \cup Y) = d_{G'}(X) + d_{G'}(Y) - 2d_{G'}(X, Y) \leq 3 + 3 - 4 = 2$ (using that both e and f connect X and Y in G'). Thus the unique 2-edge-cut in $G' - f$ is induced by X (and $Y = V - X$). Therefore it is enough to show that there exist two vertices u, v, one in X and one in Y which are not adjacent in $G' - f$ and which belong to different classes of the bipartition (A, B). The existence of such a vertex pair can be checked easily (using $|A|, |B| \geq 3$). $\square$

The above lemmas cannot be extended to $k \geq 4$. (To see this take a bipartite 4-edge-connected simple graph $\bar{G} = (A, B, E)$ and extend it by two new vertices u, v, the edge uv, two distinct edges from u to A and two distinct edges from v to B. The resulting bipartite graph is 3-edge-connected and its unique optimal augmentation wrt. $k = 4$ is a new copy of the edge uv.) Note also that the third author gave a polynomial-time algorithm which solves the special case $k = 1$ of the directed version of this bipartite augmentation problem almost optimally, that is, by finding a solution with size at most $OPT + 1$, see [12].

8 Remarks

Another type of "partition-constrained" augmentation problem is the following (let us call it Problem *PC2*).

INSTANCE *$G = (V, E)$ an undirected graph, $\mathcal{P} = \{P_1, ..., P_r\}$ a partition of V, k and M integers.*
QUESTION *Does there exist a set F of edges such that $G' = (V, E \cup F)$ is k-edge-connected, $|F| \leq M$ and the endvertices of each edge of F lie within a class of $\mathcal{P}$?*

Theorem 8.1 *Problem PC2 is NP-complete.*

Proof: The problem obviously belongs to the class NP. We reduce the Hamiltonian Cycle Problem (*HCP*) to *PC2*. Let $H = (W, E')$ be an instance of *HCP*. We can assume that H is 2-edge-connected. Let $W = \{w_1, ..., w_n\}$.

Construct the following instance of *PC2*. For every vertex $w_i \in W$ we have a vertex v_i in G and for every edge $w_iw_j \in E'$ let G contain two vertices u^1_{ij} and u^2_{ij} and two parallel edges between u^1_{ij} and v_i and two parallel edges between u^2_{ij} and v_j. Let $\mathcal{P}$ consist of the one-element classes $\{v_1\}, ..., \{v_n\}$ and the classes $\{u^1_{ij}, u^2_{ij}\}$ of size two for every pair ij with $w_iw_j \in E'$. Let $M = n$ and $k = 2$. We claim that the problem *PC2* has a solution with this instance if and only if H has a Hamiltonian cycle. The proof of this simple claim (and hence the theorem) is left to the reader. $\square$

Note that the case where r is fixed (in particular if $r = 2$) in problem *PC2* remains open.

Acknowledgments

The authors gratefully thank András Frank. He invited us to the Second Budapest Workshop on Network Design where this work was begun, and he suggested the application of Section 7. We thank Hiroshi Nagamochi for making a preprint of [18] available.

References

[1] J. Bang-Jensen and T. Jordán, Edge-connectivity augmentation preserving simplicity, Preprint no. 7, Department of Mathematics and Computer Science, Odense University (1997). To appear in the Proceedings of the Symposium on Foundations of Computer Science (FOCS) '97.

[2] E.D. Bolker and H. Crapo, How to brace a one-story building, Environ. Plan. B, 4, 125-152, 1977.

[3] G. -R. Cai and Y.-G. Sun, The minimum augmentation of any graph to a k-edge-connected graph, *Networks* **19** (1989) 151-172.

[4] E. Cheng and T. Jordán, Successive edge-connectivity augmentation problems, to appear.

[5] K.P. Eswaran and R. E. Tarjan, Augmentation problems, SIAM J. Computing 5,4, 1976, pp 653-665.

[6] A. Frank, Augmenting Graphs to Meet Edge-Connectivity Requirements, *SIAM J. Disc. Math.* **5**, 25–53 (1992).

[7] A. Frank, Connectivity augmentation problems in network design, in: Mathematical Programming: State of the Art 1994, (Eds: J.R. Birge and K.G. Murty), The University of Michigan, Ann Arbor, MI, 34-63, 1994.

[8] M. Grötschel, C.L. Monma and M. Stoer, Design of survivable networks, in: Handbook in Operations Research and Management Science Vol. 7, Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser, eds.) North-Holland, Amsterdam, 617-672, 1994.

[9] S. Hakimi, On the realizability of a set of integers as degrees of the vertices of a graph, J. SIAM Appl. Math. 10, 1962, pp. 496-506.

[10] T-S. Hsu and M-Y. Kao, Optimal augmentation for bipartite componentwise biconnectivity in linear time, Algorithms and Computation (Proc. ISAAC '96), Springer Lecture Notes in Computer Science 1178, 213-222, 1996.

[11] T. Jordán, Two NP-complete augmentation problems, Preprint no. 8, Department of Mathematics and Computer Science, Odense University (1997).

[12] T. Jordán, Ph. D. Thesis. Also: Strongly connected augmentations preserving bipartiteness, manuscript.

[13] G. Kant and H.L. Bodlaender, Planar graph augmentation problems, Algorithms and Data Structures (Proc. WADS '91), Springer Lecture Notes in Computer Science 519, 286-298, 1991.

[14] L. Lovász, Combinatorial Problems and Exercises, North-Holland, Amsterdam, 1979.

[15] W. Mader, A reduction method for edge-connectivity in graphs, Annals of Discrete Math. 3, 1978, pp. 145-164.

[16] H. Nagamochi and T. Ibaraki, Computing edge-connectivity of multigraphs and capacitated graphs, SIAM J. Discrete Math. 5, 1992, pp. 54-66.

[17] H. Nagamochi and T. Ibaraki, Deterministic $\tilde{O(}nm)$ time edge-splitting algorithm in undirected graphs, Proc. 28th Annual ACM Symp. on Theory of Computing, 1996, pp. 64-73.

[18] H. Nagamochi, S. Nakamura and T. Ibaraki, A simple $\tilde{O(}nm)$ time edge-splitting algorithm in undirected graphs, Technical Report #96013, Kyoto University, Department of Applied Mathematics and Physics, 1996.

[19] D. Naor, D. Gusfield and Ch. Martel, A fast algorithm for optimally increasing the edge-connectivity, 31st Annual Symposium on Foundations of Computer Science, 1990, pp 698-707.

[20] A. Recski, Matroid Theory and its Applications in Electric Network Theory and in Statics, Akadémiai Kiadó, Budapest, 1989.

[21] M. Stoer, Design of Survivable Networks, Ph.D. Thesis, Springer Lecture Notes in Mathematics 1531, 1992.

[22] T. Watanabe and A. Nakamura, Edge-Connectivity Augmentation Problems, *Journal of Computer and System Sciences* 35, 96–144 (1987).