

A Combinatorial Design Approach to MAXCUT

Thomas Hofmeister and Hanno Lefmann

Lehrstuhl Informatik II
Universität Dortmund
D-44221 Dortmund, Germany

Email: $\left. \begin{array}{l} \text{hofmeister} \\ \text{lefmann} \end{array} \right\} @\text{Ls2.informatik.uni-dortmund.de}$

Abstract. The k -MAXCUT problem for undirected graphs $G = (V, E)$ consists of finding a partition $V = V_1 \cup \dots \cup V_k$ such that the number of edges with endpoints in two different sets V_i is maximized. We offer a new approach to this problem by showing that the combinatorial notion of block designs can be used to algorithmically obtain partitions for which until now only existence proofs were known.

In the case of $k = 2$, we show that already known approaches can be improved by giving a simpler linear time algorithm which also yields better bounds. In particular, we give a linear time algorithm which achieves a bound of Edwards [12].

For general $k = 2^l$ and graphs with m edges, we are able to compute efficiently partitions of size $m \cdot (k - 1)/k \cdot (1 + 1/(\Delta + k - 1))$, where Δ is the maximum degree of G . The algorithms can also be applied to weighted graphs.

1 Introduction

Let $G = (V, E, w)$ be a weighted undirected graph with vertex set V , edge set E and weight function $w: E \rightarrow \mathbb{N}_0$. We will often abbreviate $n := |V|$, and $m := |E|$. The parameter Δ will denote the maximum degree of the graph. For a subgraph $G^* = (V^*, E^*)$ of G let $w(G^*) = \sum_{e \in E^*} w_e$ be the total weight of G^* . A *matching* in G is a set $\{e_1, \dots, e_r\} \subseteq E$ of pairwise non-adjacent edges.

If we have a partition $V = V_1 \cup \dots \cup V_k$ of the vertex set, we define the *cutsizes* of this partition as the sum of the weights of all crossing edges. We call an edge $e = \{v, w\}$ *crossing* if $v \in V_i$ and $w \notin V_i$ for some i . We call a partition $V = V_1 \cup \dots \cup V_k$ *balanced* if the sets V_i , $i = 1, \dots, k$, are of equal size.

Given an input graph G , the k -MAXCUT problem consists of finding a partition $V = V_1 \cup \dots \cup V_k$ where the cutsizes are as large as possible.

1.1 2-MAXCUT

It is well-known that even in the unweighted case, i.e., $w_e = 1$ for each edge $e \in E$, the problem 2-MAXCUT is NP-complete [17]. Apart from its importance from the theoretical point of view [27], 2-MAXCUT has several applications, e.g., in

the design of circuit layouts [5]. Therefore, one is interested in fast approximation algorithms for this problem.

By a probabilistic argument, Erdős proved the following:

Theorem 1. [14] *Let $G = (V, E, w)$ be a weighted graph, where $|V| = n$ is even. There exists a partition $V = V_1 \cup V_2$ such that*

$$\text{cut}(V_1, V_2) \geq \frac{w(G)}{2} \cdot \left(1 + \frac{1}{n-1}\right). \quad (1)$$

In the case of n odd, we can add a dummy vertex to the graph with all edges incident to it having weight 0. Measured in the original number of vertices, this gives a cutsize of at least $w(G)/2 \cdot (1 + 1/n)$. Observe that the lower bound (1) is sharp as the unweighted complete graph shows. Moreover, for a given unweighted graph G , Theorem 1 gives a lower bound for the largest bipartite subgraph of G . The short proof of Theorem 1 works as follows.

Proof. Choose uniformly at random a balanced partition $V = V_1 \cup V_2$. For a fixed edge $e \in E$, the probability that e is crossing is equal to $2 \cdot \binom{n-2}{n/2-1} / \binom{n}{n/2} = 1/2 \cdot (1 + 1/(n-1))$. By linearity, the expected value of the cutsize is exactly $w(G)/2 \cdot (1 + 1/(n-1))$, and, hence, there exists a partition $V = V_1 \cup V_2$ achieving at least this cutsize. \square

Erdős' proof is non-constructive, thus the natural question arises how a corresponding partition can be computed algorithmically. For connected graphs G , Poljak and Turzík gave in [25] an $O(n^3)$ algorithm which finds a partition $V = V_1 \cup V_2$ such that $\text{cut}(V_1, V_2) \geq w(G)/2 + w(T_{min})/4$, where T_{min} is a minimum-weight spanning tree in G .

For the unweighted case, we have $w(T_{min}) = n - 1$, and the existence of a partition achieving at least this cutsize was proved earlier using non-constructive arguments by Edwards [12], [13]. Later, Ngoc and Tuza [23] provided a linear time algorithm for this bound. In the weighted case, Erdős' bound can be much better.

Poljak and Turzík remarked in [25] that their strategy can be applied to find a partition with cutsize at least $w(G)/2 + w(M)/2$, where M is an arbitrary matching in G . No computational details are given. Using double-transitive subgroups of the symmetric group S_n , Poljak and Tuza [26] gave an algorithm with running time $O(n^3)$ which yields a cutsize which is not smaller than the lower bound of Erdős if n is a prime power. Using results on the density of these prime powers they obtain for arbitrary positive integers n a cutsize of at least $w(G)/2 \cdot (1 + 6/(7n))$.

Amongst other results in [18], Haglin and Venkatesan rediscovered that by using matchings, one can compute a partition with cutsize at least $m/2 + m/(2n)$. One part of their paper is devoted to a procedure which finds a matching of size at least $\lfloor m/n \rfloor$. They use this result to give an algorithm achieving at least the above cutsize with running time $O(m \log m)$. Moreover, they proved that for any fixed $\varepsilon > 0$, it is NP-complete to decide for a given graph G with m edges and a given δ , $0 < \delta < \varepsilon$, whether G contains a cut of size at least $m \cdot (1/2 + \delta)$.

Although, in principle, it would be enough to use a standard maximum matching algorithm to compute a matching of the above claimed sizes, one is interested in more efficient, i.e., linear time algorithms.

Kajitani, Cho and Sarrafzadeh presented in [20] an $O(n + m)$ time algorithm which yields a matching in G of size at least $m/(n - 1)$ for n even. To do so, they used the method of alternating augmenting paths. This algorithm is extended in [20] in a rather involved way to the bound $w(G)/(n - 1)$ for weighted graphs. Based on this, Cho, Raje and Sarrafzadeh gave in [11] a linear time algorithm which finds a partition with cutsize at least $m/2 \cdot (1 + 1/n)$. For weighted graphs, they obtain a cutsize of at least $w(G)/2$.

As it looks from the existing literature, for n even, it seems to have been overlooked that finding a matching of weight at least $w(G)/(n - 1)$ in linear time can be achieved more easily. Using this and the algorithm which has been discovered a few times, one gets a simple and fast $O(n + m)$ algorithm for finding a partition with cutsize at least $w(G)/2 \cdot (1 + 1/(n - 1))$ in weighted graphs for n even. Using proper vertex-colorings, this can be improved to the value $w(G)/2 \cdot (1 + 1/\Delta)$ for Δ odd. Moreover, we give a linear-time algorithm for a result of Edwards [12] on the largest size of a bipartite subgraph.

1.2 k -MAXCUT

By an argument similar to the one used in the proof of Theorem 1, Andersen, Grant and Linial [4] showed that for general k , there is always a partition with cutsize at least $m \cdot (k - 1)/k \cdot (1 + 1/(n - 1))$, if n is divisible by k . In their argument they used the uniform distribution on subsets of fixed size. We consider the problem of how this lower bound can be obtained algorithmically for weighted graphs. It turns out that for some special values of k , a simple divide-and-conquer strategy will do. For other values of k , we reveal that the concept of *balanced incomplete block designs* combined with derandomization of the corresponding binomial distribution is a suitable approach to the problem. These results improve upon recent results in [11].

Again, it should also be noted that Ngoc and Tuza [23] gave linear time algorithms which compute in the unweighted case and for connected graphs a partition with cutsize at least $m \cdot (k - 1)/k + (n - 1)/k$, if $k \geq 3$. They also showed that for every real $\varepsilon > 0$, and for every integer $k \geq 2$ with $\varepsilon < 1/(k - 1)$ it is NP-complete to decide whether a graph admits a k -cut with cutsize at least $(k - 1)/k \cdot m \cdot (1 + \varepsilon)$. Thus, $(k - 1)/k \cdot m$ is essentially the borderline for polynomial time algorithms, unless $P = NP$.

2 2-MAXCUT

2.1 1-Factorizations

In order to turn Erdős' existence proof into a deterministic procedure, the following lemma has been discovered a number of times. It has also been observed

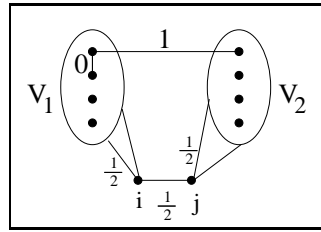
that instead of matchings we can use a list of pairwise vertex disjoint induced bipartite subgraphs, cf. [27]. As the underlying procedure will be analogous to the procedure that we apply when dealing with k -cuts, we provide the proof and the algorithm. The algorithm can be seen as a derandomized version of Erdős' argument using the method of conditional probabilities, cf. [2]. However, instead of the uniform distribution (which would result in a non-polynomial time algorithm), the binomial distribution together with matchings is used.

Lemma 2. *Let $G = (V, E, w)$ be a weighted graph. Given a matching M of G , there is an algorithm with running time $O(n + m)$ which finds a partition $V = V_1 \cup V_2$ such that $\text{cut}(V_1, V_2) \geq \frac{w(G) + w(M)}{2}$. If n is even, the constructed partition is balanced.*

Proof. We start by initializing $V_1 := V_2 := \emptyset$. During the algorithm, we maintain a parameter $\text{Val}(e)$ for every edge $e = \{i, j\}$ and a potential function $\text{VAL}(G)$ for the whole graph. They are defined as follows:

$$\text{Val}(e) = \begin{cases} w_e/2 & \text{if } |e \cap (V_1 \cup V_2)| \leq 1 \\ w_e & \text{if } |e \cap V_1| = |e \cap V_2| = 1 \\ 0 & \text{otherwise.} \end{cases} \quad \text{VAL}(G) = \sum_{e \in E} \text{Val}(e)$$

The figure below gives a sketch of the situation, where on the edges we indicated the coefficients of the corresponding weights. Initially, $\text{VAL}(G) = w(G)/2$. We process the edges of the matching M one after the other.



Given an edge $e = \{i, j\} \in M$ which fulfills $e \cap (V_1 \cup V_2) = \emptyset$, we decide to either add i to V_1 and j to V_2 , or we decide to add j to V_1 and i to V_2 . In both cases, $\text{Val}(e)$ changes from $w_e/2$ to w_e . The parameters of some other edges change as well, namely, of all edges which have one vertex in $\{i, j\}$ and the other in $V_1 \cup V_2$. Call this edge set E' .

For $e' \in E'$, the value of $\text{Val}(e')$ changes from $w_{e'}/2$ to either $w_{e'}$ or 0, depending on which of the two possibilities we choose. By the pigeon-hole principle, one choice does not decrease the sum of all $\text{Val}(e')$, $e' \in E'$. Thus, we can increase $\text{Val}(e)$ whilst not decreasing $\sum_{e' \in E'} \text{Val}(e')$. We repeat the above procedure for all edges in the matching M . We then have $\text{VAL}(G) \geq (w(G) + w(M))/2$.

If some vertices remain to be distributed over V_1 and V_2 , then we can group them into pairs and continue as above without reducing $\text{VAL}(G)$. If n is odd, we add a dummy vertex.

Finally, we obtain a partition $V = V_1 \cup V_2$. By definition, then $\text{VAL}(G) = \text{cut}(V_1, V_2) \geq (w(G) + w(M))/2$. As every edge is examined only a constant number of times, the running time is $O(n + m)$. \square

Lemma 2 yields an algorithm for computing a partition which has a cutsize achieving at least Erdős' bound if we can come up with a matching M of weight $w(M) \geq w(G)/(n-1)$. Actually, it is easy to compute such a matching. Namely, a 1-factorization yields a linear time algorithm which gives a matching of size at least $w(G)/(n-1)$:

Lemma 3. *Let $G = (V, E, w)$ be a weighted graph with n even. One can compute in time $O(n + m)$ a matching M in G which has weight $w(M) \geq w(G)/(n - 1)$.*

Proof. We use the fact that the complete graph K_n has a 1-factorization for even n , i.e., its edge set can be partitioned into $n - 1$ matchings of $n/2$ edges each. Namely, for $V = \{1, \dots, n\}$, the matchings $M_j = \{\{j, n\}\} \cup \{\{j - i, j + i\} \mid i = 1, \dots, \frac{n}{2} - 1\}$, $j = 1, \dots, n - 1$, yield a 1-factorization, where addition is done cyclically in $\{1, \dots, n - 1\}$.

As $w(G) = \sum_{j=1}^{n-1} w(M_j)$, there exists a matching M_j with weight $w(M_j) \geq w(G)/(n - 1)$. Observe that for every edge $e \in E$, it is straightforward to compute in time $O(1)$ the matching M_j this edge e belongs to. By processing all edges we can compute $w(M_j)$ for all j in time $O(n + m)$. Hence, if we choose the matching M_j with maximum weight, we can guarantee that $w(M_j) \geq w(G)/(n - 1)$. \square

Combining Lemma 2 and Lemma 3, we get a linear time algorithm for computing a partition which achieves at least the weight which is guaranteed by Erdős' theorem.

2.2 Improvements by Colorings

Here we show that Erdős' theorem can be strengthened by using colorings. A linear time algorithm for the strengthened version is still available.

Let us first recall the definitions of colorings: A *proper edge-coloring* of $G = (V, E, w)$ with t colors is a function $c: E \rightarrow \{1, \dots, t\}$ such that for all adjacent edges $e_1 \neq e_2$ in E , we have $c(e_1) \neq c(e_2)$. The *chromatic index* $\chi_e(G)$ is the minimum number t such that there is a proper edge-coloring with t colors. In an analogous fashion, the notions of a *proper vertex-coloring* and the *vertex chromatic number* χ_v can be defined. Note that computing the vertex chromatic number, as well as computing the chromatic index, is known to be NP-complete. The considerations of Subsection 2.1 can be strengthened by taking into account the maximum degree of a graph.

Given a proper edge coloring, all edges with the same color form a matching. If t colors suffice, then one of those matchings has size at least $w(G)/t$.

A theorem of Vizing [29] gives a sharp estimate on the chromatic index of a graph in terms of its maximum degree, namely $\Delta \leq \chi_e(G) \leq \Delta + 1$. Using the above considerations, Lemma 2 yields a better guaranteed cutsize. A polynomial time algorithm for properly coloring edges with at most $\Delta + 1$ colors has been given by Misra and Gries [22]. The running time of their algorithm is not linear. Nevertheless, we will see that in almost all cases, it is better to use vertex-colorings.

Observation 4 *Given a proper vertex-coloring with t colors, each nonempty color class $c^{-1}(i)$, $i = 1, \dots, t$ is an independent set in G . We collapse each such color class $c^{-1}(i)$ to a single vertex v_i and consider the new weighted graph $G' = (V', E', w')$ with $V' = \{v'_1, \dots, v'_t\}$, where for $e' = \{v'_i, v'_j\} \in E'$ it is*

$$w'_{e'} = \sum_{e=\{r,s\} \in E; r \in c^{-1}(i), s \in c^{-1}(j)} w_e,$$

Afterwards, we can apply the algorithm from Subsection 2.1 to the graph G' . We obtain a partition $V = V_1 \cup V_2$ such that $\text{cut}(V_1, V_2) \geq \frac{w(G)}{2} \cdot (1 + \frac{1}{t-1})$ if t is even.

We remind the reader of the following properties of vertex-colorings (see e.g. [7]):

Proposition 5. *A graph $G = (V, E)$ with maximum degree Δ has a proper vertex-coloring with at most $\Delta + 1$ colors. Such a coloring can be computed in time $O(n + m)$.*

Proposition 6. *Let $G = (V, E)$ and define $t := \max\{k \mid \binom{k}{2} \leq m\}$. Then G has a proper vertex-coloring with at most t colors. Such a coloring can be computed in time $O(n + m)$.*

If we combine our linear-time algorithm behind Observation 4 with the algorithm behind Proposition 5, we obtain the following:

Theorem 7. *Let $G = (V, E, w)$ be a weighted graph with odd maximum degree Δ . There is an algorithm with running time $O(n + m)$ which computes a partition $V = V_1 \cup V_2$ such that $\text{cut}(V_1, V_2) \geq \frac{w(G)}{2} \cdot (1 + \frac{1}{\Delta})$.*

Using Proposition 6, we obtain the following which is known as Edwards' bound and which first was proved in [12], [13] in a somewhat involved fashion.

Theorem 8. *For every graph $G = (V, E)$, there is a partition $V = V_1 \cup V_2$ with $\text{cut}(V_1, V_2) \geq m/2 + (\sqrt{8m + 1} - 1)/8$. Such a partition can be computed in time $O(n + m)$.*

Proof. The parameter t in Proposition 6 fulfills $t \leq 1/2 + \sqrt{2m + 1}/4$. Using the algorithm from Observation 4 we obtain a cut with the claimed size if t is even. If t is odd, we add a dummy vertex. \square

Notice that for weighted graphs $G = (V, E, w)$ the lower bound on $\text{cut}(V_1, V_2)$ in Theorem 8 can be replaced by $w(G)/2 \cdot (1 + (\sqrt{8m + 1} - 1)/(4m))$.

Poljak and Tuza [27] also found a short proof of Edwards' bound but did not give a linear-time algorithm. Recently, Erdős, Gyárfás and Kohayakawa [16], found another short proof of Edwards' bound. Alon [1] proved that there are constants $c > 0$ and l_0 such that for all even $l \geq l_0$, and any m of the form $m = l^2/2$, Edwards' bound can be improved by an additive term of $cm^{1/4}$ for every graph with m edges. This recent interest in Edwards' bound was incited by a talk of Erdős [15].

We remark that by a theorem of Brooks [10], every connected graph which is neither the complete graph nor an odd cycle satisfies $\chi_v(G) \leq \Delta$. For references on efficient algorithmic versions of Brooks' theorem we refer to Karloff [19].

3 k -MAXCUT

3.1 k -MAXCUT for $k = 2^l$

Cho, Raje and Sarrafzadeh gave in [11] an algorithm with running time $O(n \log k + m)$ which finds for every graph on n vertices and with m edges a partition of

the vertex set into k subsets with cutsize at least

$$m \cdot \left(1 - \frac{1}{k} \cdot \left(\frac{n-1}{n}\right)^{\lceil \log_2 k \rceil}\right). \quad (2)$$

However, the divide-and-conquer strategy yields a better guaranteed k -cutsize:

Theorem 9. *Let $k = 2^l$, where $l \in \mathbb{N}$ and let $G = (V, E, w)$ be a weighted graph with n divisible by k . There is an algorithm with running time $O((n+m) \log k)$ which computes a partition $V = V_1 \cup \dots \cup V_k$ such that*

$$\text{cut}(V_1, \dots, V_k) \geq w(G) \cdot \frac{k-1}{k} \cdot \left(1 + \frac{1}{n-1}\right). \quad (3)$$

In the unweighted case the algorithm has running time $O(n \log k + m)$.

Proof. The idea for the algorithm is simply to iterate the 2-cut algorithm. First, we obtain a partition $V = V_1 \cup V_2$ with $|V_1| = |V_2|$ such that $\text{cut}(V_1, V_2) \geq w(G)/2 \cdot (1 + 1/(n-1))$. Then we apply the maxcut algorithm from Subsection 2.1 to each subgraph of G induced by V_i , $i = 1, 2$, and so on.

Define the *insize* of a partition $V = V_1 \cup \dots \cup V_k$ by $\text{in}(V_1, \dots, V_k) = w(G) - \text{cut}(V_1, \dots, V_k)$. It is enough to prove that the above strategy yields a partition $V = V_1 \cup \dots \cup V_k$ such that

$$\text{in}(V_1, \dots, V_k) \leq \frac{w(G)}{k} \cdot \left(1 - \frac{k-1}{n-1}\right). \quad (4)$$

Inequality (4) follows by observing that after i rounds, each set in the partition consists of $n/2^i$ vertices which means that after $l = \log_2 k$ rounds,

$$\begin{aligned} \text{in}(V_1, \dots, V_k) &\leq w(G) \cdot \prod_{i=0}^{l-1} \frac{1}{2} \cdot \left(1 - \frac{1}{\frac{n}{2^i} - 1}\right) = \frac{w(G)}{2^l} \cdot \prod_{i=0}^{l-1} \left(\frac{n - 2^{i+1}}{n - 2^i}\right) \\ &= \frac{w(G)}{2^l} \cdot \frac{n - 2^l}{n - 1} = \frac{w(G)}{k} \cdot \frac{n - k}{n - 1}. \end{aligned}$$

As the running time for the 2-maxcut algorithm in each round is linear in $(n+m)$, we conclude that the procedure described above has a running time of $O((n+m) \log k)$. Moreover, in the unweighted case the running time drops to $O(n \log k + m)$, as the number of edges considered in round i is bounded from above by $m/2^{i-1}$. \square

The lower bound (3) is larger than (2). Namely, let $w_e = 1$ for all $e \in E$. First notice that for $k \geq 2$ and $n \geq 2$ the following inequality holds: $\frac{k-1}{n-1} - \frac{\lceil \log_2 k \rceil}{n} > 0$. Then, using $(1-x)^l \geq 1-lx$, we infer $\left(\frac{n-1}{n}\right)^{\lceil \log_2 k \rceil} - 1 + \frac{k-1}{n-1} > 0$, which is equivalent to $\frac{k-1}{k} \left(1 + \frac{1}{n-1}\right) > 1 - \frac{1}{k} \cdot \left(\frac{n-1}{n}\right)^{\lceil \log_2 k \rceil}$, as claimed.

We remark that the bound (3) is tight, as the complete graph K_n shows.

Again, we get better results by using vertex colorings. Given a graph with maximum degree Δ , we can apply the vertex collapsing strategy from Section 2.2. If $\Delta + 1$ is not divisible by k , we can add at most $k - 1$ dummy vertices and obtain the following:

Theorem 10. *Let $G = (V, E, w)$ be a weighted graph with maximum degree Δ . Let $k = 2^l$, where $l \in \mathbb{N}$. There is an algorithm with running time $O((n + m) \cdot \log k)$ which computes a partition $V = V_1 \cup \dots \cup V_k$ such that $\text{cut}(V_1, \dots, V_k) \geq w(G) \cdot \frac{k-1}{k} \cdot \left(1 + \frac{1}{\Delta+k-1}\right)$.*

We only remark that our considerations yield also improvements for the maximum k -covering problem given in [11].

3.2 k -MAXCUT and Resolvable Block Designs

We first recall some basic notions from combinatorial design theory.

Definition 11. *A balanced incomplete block design, BIBD, with parameters (n, k, λ) , is a list B_1, \dots, B_r of k -subsets from $\{1, \dots, n\}$ such that every 2-subset $A \subseteq \{1, \dots, n\}$ is contained in exactly λ sets B_i . The subsets B_i are called *blocks*. In particular, a $(n, 3, 1)$ -BIBD is called a *Steiner triple system*. A BIBD is *resolvable* if its blocks can be arranged into groups such that every group is a partition of $\{1, \dots, n\}$. A resolvable Steiner triple system is called a *Kirkman triple system*.*

Clearly, each (n, k, λ) -BIBD consists of $\lambda \binom{n}{2} / \binom{k}{2}$ blocks. The following is an example of a Steiner triple system with $n = 9$:

$$\begin{aligned} &\{1, 2, 3\}, \{4, 8, 9\}, \{5, 6, 7\}, \quad \{1, 5, 8\}, \{3, 4, 6\}, \{2, 7, 9\}, \\ &\{1, 4, 7\}, \{2, 6, 8\}, \{3, 5, 9\}, \quad \{1, 6, 9\}, \{2, 4, 5\}, \{3, 7, 8\}. \end{aligned}$$

Indeed, the example yields a Kirkman triple system consisting of 4 groups. The existence of Kirkman triple systems for all $n \equiv 3 \pmod{6}$ was proved by Ray-Chaudhuri and Wilson [28]. There is a variety of construction methods known for BIBDs, cf. Anderson [3].

Resolvable $(n, 2, 1)$ -BIBDs are well-known in graph theory, namely, they yield a 1-factorization of the complete graph. Hence, Subsection 2.1 shows why BIBDs can be useful for computing large cutsizes in a graph. The method sketched there can be generalized to resolvable (n, k, λ) -BIBDs for finding k -cuts.

Lemma 12. *Let $G = (V, E, w)$ be a weighted graph with $|V| = n$. Assume that we are given a resolvable (n, k, λ) -BIBD which consists of the blocks B_1, \dots, B_r and assume that they are already arranged in groups, i.e., B_1, \dots, B_t belong to group 1, and B_{t+1}, \dots, B_{2t} belong to group 2 etc. Then we can compute in time $O(k(n + m) + \lambda n^2)$ a balanced partition $V = V_1 \cup \dots \cup V_k$ such that its cutsize is at least $w(G) \cdot \frac{k-1}{k} \cdot \left(1 + \frac{1}{n-1}\right)$.*

Proof. Every block B_i can be seen as a set of vertices of G and we can associate the subgraph $G(B_i)$ of G induced by B_i with it. In a first run-through, we compute the weights $w(B_i) := w(G(B_i))$ for $i = 1, \dots, r$ and the weight of every group (where the weight of a group is the sum of the weights of its blocks).

Now we make use of the properties of a (n, k, λ) -BIBD. Since every edge $e \in E$ is contained in exactly λ of the blocks, we have $w(B_1) + \dots + w(B_r) = \lambda \cdot w(G)$. The BIBD is arranged into $r/(n/k)$ groups each containing n/k blocks. Thus, one of the groups must have weight at least $\lambda \cdot w(G) \cdot n/(kr) = w(G) \cdot (k-1)/(n-1)$, as $r = \lambda \binom{n}{2} / \binom{k}{2}$. Finding such a group takes time $O(\lambda n^2)$. W.l.o.g. we assume that this group consists of the blocks B_1, \dots, B_t .

As we did for 2-cuts, we initialize $V_1 := \dots := V_k := \emptyset$. During the algorithm we keep track of edge parameters $Val(e)$ which are defined as follows:

$$Val(e) = \begin{cases} w_e & \text{if the vertices of } e \text{ are in different sets } V_i, \\ 0 & \text{if } e \subseteq V_i \text{ for some } i, \\ \frac{k-1}{k} \cdot w_e & \text{otherwise.} \end{cases}$$

Moreover, we have a potential $VAL(G) = \sum_{e \in E} Val(e)$. The algorithm processes the blocks B_1, \dots, B_t in succession, adding their vertices to the sets V_i . This is possible since no two blocks of a group have a vertex in common.

Assume that we arrive at a point where we process block B_i , say $B_i = \{v_1, \dots, v_k\}$. We now want to decide between k alternatives. Namely, choosing the l th possibility means that we place v_1 into V_l , v_2 into V_{l+1} etc. (where the addition is done cyclically). This means that we add the vertices cyclically to the sets V_j , we only have to choose the ‘‘anchor point’’ for v_1 . No two vertices of B_i are placed into the same set V_j , hence for all edges e in $G(B_i)$ we have that $Val(e)$ changes from $\frac{k-1}{k} \cdot w_e$ to w_e .

$Val(e')$ also changes for edges e' which have one endpoint in B_i and the other in $V_1 \cup \dots \cup V_k$. Let this set of edges be called E' . For an edge $e' \in E'$, amongst the k placing possibilities that we have to choose from, exactly $(k-1)$ will change $Val(e')$ from $\frac{k-1}{k} \cdot w_{e'}$ to $w_{e'}$ and only one possibility will change $Val(e')$ to 0. There must be one possibility which does not decrease the contribution of E' to $VAL(G)$. Thus, we can increase $Val(e)$ for each edge in $G(B_i)$ whilst not decreasing $\sum_{e' \in E'} Val(e')$.

By the above arguments, in each such step the value $VAL(G)$ is increased by at least $w(B_i)/k$. Since we started off the algorithm with $VAL(G) = w(G) \cdot (k-1)/k$ and added the above amount for every block B_i processed, we finish with $VAL(G) = w(G) \cdot (k-1)/k + \sum_{i=1}^t w(B_i)/k$, which by our choice of the group is at least $w(G) \cdot (k-1)/k \cdot (1 + 1/(n-1))$. Since a group covers all vertices, all vertices are distributed at the end of the algorithm and $VAL(G)$ gives the actual cutsizes.

Clearly, the placing takes time $O(k(m+n))$ which means that we get an overall running time of $O(k(n+m) + \lambda n^2)$. \square

It should be noted that the choice of λ only influences the running time and not the guaranteed cutsizes of the partition. Also, if the BIBD is given in such a way that for every edge e we can compute in time $O(\lambda)$ the λ groups this edge

belongs to, then the term λn^2 can be replaced by λm in the running time using an appropriate data structure.

Although various algebraic construction methods for BIBDs are known in the literature, see e.g. [3], not a lot of attention has been paid to their exact running times. It might be interesting to investigate this more closely.

In the following, we discuss just a few of the existing constructions. Let us start with the following theorem by Baranyai.

Theorem 13. [6] *Let k, n be positive integers, where n is divisible by k . Let $X := \{1, \dots, n\}$. Then the set $[X]^k$ of k -element subsets of X can be partitioned into $r = \binom{n-1}{k-1}$ sets X_1, \dots, X_r such that for $i = 1, \dots, r$ and distinct sets $S, T \in X_i$ it is $S \cap T = \emptyset$.*

Theorem 13 shows the existence of a 1-factorization for the k -uniform complete hypergraph K_n^k , if n is divisible by k . In terms of BIBDs, this can be read as follows: The set system $[\{1, \dots, n\}]^k$ is a resolvable $(n, k, \binom{n-2}{k-2})$ -BIBD.

Theorem 13 is not suitable for our application, namely for two reasons: The first one is that the original proof does not provide us with a polynomial time procedure for arranging the blocks into groups. Second, even if we were given this arrangement for free, then it should be observed that the value of λ in the construction is rather large, so that the running time of our algorithm in Lemma 12 would be a polynomial of a large degree.

For $k = 3$ and a large number of values of n , Beth gave a polynomial time algorithm for obtaining an arrangement of the triples into groups as guaranteed by Theorem 13.

Theorem 14. [8] *Let n be a positive integer such that n is divisible by 3, and either n or $n-1$ or $n-2$ is a prime power. There is a polynomial time algorithm to construct a 1-factorization of the complete 3-uniform hypergraph K_n^3 .*

For further values of n more complicated constructions have been given by Peltesohn [24] in her dissertation, where Theorem 13 for $k = 3$ is proved. However, it seems that the applied techniques do not lead to fast algorithms, cf. [8].

Using the result of Beth, Lemma 12 yields the following

Proposition 15. *Let n be a positive integer divisible by 3, such that n , $n-1$ or $n-2$ is a prime power. Let $G = (V, E, w)$ be a weighted graph on n vertices. There is a polynomial time algorithm which yields a partition $V = V_1 \cup V_2 \cup V_3$ such that $\text{cut}(V_1, V_2, V_3) \geq w(G) \cdot \frac{2}{3} \cdot \left(1 + \frac{1}{n-1}\right)$.*

By a result of Breusch [9] for every integer $m \geq 48$ there exists a prime p with $m \leq p \leq 9/8m$. Thus for every integer $m \geq 48$ there exists an integer $n \equiv 0 \pmod{3}$ such that $m \leq n \leq 7/6m$ and n , $n-1$ or $n-2$ is a prime power. As in [26], checking the integers $m \leq 47$, we obtain that for every integer $m \geq 6$ there is an $n \equiv 0 \pmod{3}$ with $m \leq n \leq 9/7m$ such that n , $n-1$ or $n-2$ is a prime power. Hence, for every graph G on n vertices, $n \geq 6$, we can construct a 3-cut

of size at least $w(G) \cdot 2/3 \cdot (1 + 7/(9n - 7))$. However, due to the density of the primes one can construct a 3-cut of size at least $w(G) \cdot 2/3 \cdot (1 + (1 - o(1))/n)$ for n large.

For $k = 3$, Kirkman triple systems could also be used; the constructions for such systems seem to be more efficient; nevertheless, the corresponding running times have not been considered so far. We mention the following example of a Kirkman triple system. For n being a power of 3, $n = 3^l$, the underlying set consists of all affine points in the vector space $(GF(3))^l$, and the blocks contain the points of the affine lines in $(GF(3))^l$. Using the Schubert normal form of matrices one can group the blocks according to resolvability in time $O(n^2)$. Thus, for n being a power of 3, we obtain in running time $O(n^2)$ a 3-cut of size at least $w(G) \cdot 2/3 \cdot (1 + 1/(n - 1))$ by Lemma 12. Clearly, for a fixed prime power q and $n = q^l$, with a similar construction - considering the affine space $(GF(q))^l$ - one can obtain in time $O(n^2)$ a q -cut with cutsize as least as guaranteed by the probabilistic argument.

Also, the following should be noted. In [21], Lu proved that for every k and λ , there is an n_0 such that for all $n \geq n_0$, there is a resolvable (n, k, λ) -BIBD if the trivial conditions for k and λ are fulfilled, cf. [3] p. 203. For $\lambda = k - 1$, these trivial conditions reduce to $n \equiv 0 \pmod k$. The question remains how we can compute such a resolvable BIBD by an efficient algorithm.

Finally, we want to remark that by Observation 4 the vertex number n within the bounds of Section 3.2 can be replaced by the vertex chromatic number, yielding better guaranteed cutsizes in many cases.

References

- [1] N. Alon, Bipartite Subgraphs, preprint, 1995.
- [2] N. Alon and J. Spencer, The Probabilistic Method, Wiley & Sons, New York, 1992.
- [3] I. Anderson, Combinatorial Designs, Construction Methods, Ellis Horwood, 1990.
- [4] L. Andersen, D. Grant and N. Linial, Extremal k -colourable Subgraphs, Ars Comb. 16, 1983, 259-270.
- [5] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design, Operations Research 36, 1988, 493-513.
- [6] Z. Baranyai, On the Factorization of the Complete Uniform Hypergraph, in: Finite and Infinite Sets, eds. Hajnal, Rado, Sós, Coll. Math. Soc. János Bolyai 10, North Holland, Amsterdam, 1973, 91-107.
- [7] C. Berge, Graphs, North-Holland, 1985.
- [8] T. Beth, Algebraische Auflösungsalgorithmen für einige unendliche Familien von 3-Designs, Le Math. 29, 1974, 105-135.
- [9] R. Breusch, Zur Verallgemeinerung des Bertrand'schen Postulates, dass zwischen x und $2x$ stets Primzahlen liegen, Math. Z. 34, 1931, 505-526.
- [10] R. Brooks, On Coloring the Nodes of a Network, Camb. Phil. Soc. 37, 1941, 194-197.

- [11] J. D. Cho, S. Raje and M. Sarrafzadeh, Approximation Algorithm on Multi-Way Maxcut Partitioning, Proc. 2nd Europ. Symp. on Algorithms, Springer LNCS 855, 1994, 148-158.
- [12] C. S. Edwards, Some Extremal Properties of Bipartite Graphs, Can. J. Math. 25, 1973, 475-485.
- [13] C. S. Edwards, An Improved Lower Bound for the Number of Edges in a Largest Bipartite Subgraph, in: Recent Advances in Graph Theory, Academia, Prag, 1975, 167-181.
- [14] P. Erdős, On Bipartite Subgraphs of Graphs, Math. Lapok 18, 1967, 283-288.
- [15] P. Erdős, Some Recent Problems in Combinatorics and Graph Theory, preprint, 1995; and Lecture at the 26th Southeastern Int. Conf. on Graph Theory, Combinatorics and Computing, Boca Raton, 1995.
- [16] P. Erdős, A. Gyárfás and Y. Kohayakawa, The Size of the Largest Bipartite Subgraphs, preprint, 1995.
- [17] M. R. Garey, D. S. Johnson, and L. Stockmeyer, Some Simplified NP-complete Graph Problems, Theo. Comp. Sci. 1, 1976, 237-267.
- [18] D. J. Haglin and S. M. Venkatesan, Approximation and Intractability Results for the Maximum Cut Problem and its Variants, IEEE Trans. Comp. 40, 1991, 110-113.
- [19] H. Karloff, An NC Algorithm for Brooks' Theorem, Theo. Comp. Sci. 68, 1989, 89-103.
- [20] Y. Kajitani, J. D. Cho, and M. Sarrafzadeh, New Approximation Results on Graph Matching and Related Problems, Proc. 20th Workshop on Graph Theoretic Concepts in Computer Science, Springer LNCS 903, 1994, 343-358.
- [21] J. X. Lu, An Existence Theorem for Resolvable Balanced Incomplete Block Designs, (in Chinese), Acta Math. Sin. 27, 1984, 458-468.
- [22] J. Misra and D. Gries, A Constructive Version of Vizing's Theorem, Inf. Proc. Letters 41, 1992, 131-133.
- [23] N. van Ngoc and Z. Tuza, Linear Time Approximation Algorithms for the Max Cut Problem, Comb., Prob. Comp. 2, 1993, 201-210.
- [24] R. Pelsesohn, Das Turnierproblem für Spiele zu je dreien, Dissertation, Berlin, 1936.
- [25] S. Poljak and D. Turzík, A Polynomial Time Heuristic for Certain Subgraph Optimization Problems with Guaranteed Worst Case Bound, Disc. Math. 58, 1986, 99-104.
- [26] S. Poljak and Z. Tuza, Bipartite Subgraphs of Triangle-Free Graphs, SIAM J. Disc. Math. 7, 1994, 307-313.
- [27] S. Poljak and Z. Tuza, Maximum Cuts and Largest Bipartite Subgraphs, in: Combinatorial Optimization, eds. Cook, Lovász, Seymour, AMS, 1995, 181-244.
- [28] D. K. Ray-Chaudhuri and R. M. Wilson, Solution of Kirkman's Schoolgirl Problem, Proc. Symp. Math. 19, 1971, 187-203.
- [29] V. G. Vizing, On an Estimate of the Chromatic Class of a p -Graph, (in Russian), Diskret. Analiz 3, 1964, 23-30.