Approximate Max k-Cut with Subgraph Guarantee

Viggo Kann[1*], Jens Lagergren[1**], Alessandro Panconesi[2***]

[1] Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-100 44 Stockholm, Sweden. Email: {jensl,viggo}@nada.kth.se

[2] Facbereich Informatik, Freie Universität, Takustr. 9, 14195 Berlin, Deutschland. Email: ale@inf.fu-berlin.de

Abstract. We study the following variant of the Max k-Cut problem. Given an input graph G with positively weighted edges and k colors– the number k being fixed and not dependent on the input instance– we wish to compute a subgraph H of G containing "lots" of heavy edges and a color assignment $c : V \to [k]$ such that: (a) all edges in H are properly colored and (b) a "large fraction" of edges in $G \setminus H$ is properly colored. We give several definitions of "lots" and "large fraction" and give fast polynomial time algorithms to compute such color assignments. This problem is related to the frequency allocation problems for cellular telephone networks but could be useful in other scenarios too.

1 Introduction.

In the *frequency allocation problem for cellular telephones* we are given a set of k frequencies, henceforth referred to as *colors*, and a weighted input graph which intuitively models a network of radio transmitters. In the graph, vertices correspond to transmitters, and an edge (u, v) denotes the fact that if u and v get the same color (*i.e.* frequency) the performance locally deteriorates by $w(u, v)$ per cent. Usually, the set of colors is fixed once for all and it is independent of the input graph. The goal is to allocate colors so to minimize the "total disruption", a notion that can be formalized in several ways. In many situations one is seeking to optimize simultaneously two somewhat conflicting parameters. On one hand, one would like to take care of "heavy" edges as much as possible because if even one of these was monochromatic, the resulting deterioration of local service would be unacceptable (an edge is *monochromatic* if both endpoints have the same color). On the other, one would like the color assignment to be good globally, *i.e.* the fraction of monochromatic edges should be small.

One way to formalize the problem is to introduce a parameter h and to define the heavy edges as those whose weight is greater than or equal to h. This

* Supported by grants from TFR.

** Supported by grants from NFR and TFR.

*** Supported by an ERCIM post-doctoral fellowship and by an Alexander von Humboldt fellowship.

formulation however immediately leads to an NP-hard problem because asking whether such a subgraph is k-colorable is an NP-complete problem. Therefore other approaches are called for. In this paper we propose two ways to define the graph of "heavy" edges which do not give raise to an NP-complete problem and that could work well in many real life situations. The basic idea is that we are given a subgraph H of G which: (a) is easily computable; (b) is easily k-colorable, and (c) contains lots of "heavy" edges (hopefully all). Our goal is then to compute a color assignment of G that satisfies these two conditions simultaneously: (1) no edge of H is monochromatic; (2) only a "small" fraction of the remaining edges is monochromatic.

In this paper we show how the above goal can be achieved for two categories of graphs: maximum l-matchings (to be defined in section 2) and unions of maximum weight spanning trees. We will investigate the following scenarios.

- H is a maximum $(k-1)$-matching in G. In this case the fraction of monochromatic edges in $G \backslash H$ is at most $2\mathcal{H}(k)/k$, where $\mathcal{H}(n)$ is the harmonic number. This can be achieved in $O(k^5|E| + k|V|)$ deterministic sequential time.
- H is a maximum $(k/2)$-matching in G. In this case the fraction of monochromatic edges in $G \backslash H$ is at most $2/k$. This can be achieved in $O(|E|+|V|k \log k)$ deterministic sequential time.
- H is a maximum $(\log k - 1)$-matching in G. In this case the fraction of monochromatic edges in $G \setminus H$ is at most $1/k$. This can be achieved in $O((|V| + |E|) \log k)$ deterministic sequential time.
- H is the union of t maximum weight spanning trees (MST's). In this case the fraction of monochromatic edges in $G \setminus H$ is at most $1/(k - 2t + 1)$. This can be achieved in $O(|E| + |V|k \log k)$ deterministic sequential time.

How good are these solutions? To have an idea consider the MAX k-CUT problem: given an input graph G with weighted edges, find a color assignment that maximizes the total weight of non monochromatic edges. The best known approximation algorithm computes color assignments such that the fraction of monochromatic edges is $(1 + o(1))/k$ [6, 2]. On the other hand, in a recent paper Kann et al. [3] show that no polynomial time algorithm can guarantee a fraction of monochromatic edges smaller than $1/\sigma k$, unless P=NP. Here σ is a constant depending on the best known lower-bound for MAX CUT whose current value is 132. This lower bound automatically applies to our problems (for which higher lower bounds could perhaps be found).

Comment: we are assuming that random numbers can be obtained for free.

2 Preliminaries

We will use the following notation.

If G is a graph, then $\Delta(G)$ denotes the maximum degree of any vertex of G.

A *maximum l-matching* in G is a subgraph $H \subseteq G$ with $\Delta(G) \leq l$ that has maximum weight among such subgraphs [4].

A graph G is *d-inductive* if (1) it has a single vertex or (2) there is a vertex of degree at most d in G such that $G \setminus \{v\}$ is d-inductive.

Let H and H' be subgraphs of G. Then, $G \setminus H$ is the graph with vertex set $V(G)$ and edge set $E(G) \setminus E(H)$. Similarly, $H \cup H'$ is the graph with vertex set $V(G)$ and edge set $E(H) \cup E(H')$.

3 Maximum matchings

3.1 Many guaranteed edges

In this subsection we give an algorithm which, given a weighted input graph G and a maximum k-matching H in G, finds a k-coloring $c : V \to [k]$ of G such that (a) no edge of H is monochromatic, and (b) at most a $2\mathcal{H}(k)/k$ fraction of edges in $G \setminus H$ is monochromatic.

We need a preliminary definition.

Definition 1. A coloring $c : V \to [\ell]$ of a a graph H is a *good coloring* if, for all vertices u, no two neighbors of u have the same color.

Given a graph H, a good coloring using $\ell = \Delta(H)^2 + 1$ colors can be easily computed as follows. Consider the graph H^2 which is the same graph as H with edges added between any two vertices u and v which, in H, have distance two or less. We have that $\Delta(H^2) \leq \Delta(H)^2$. Any coloring of H^2 is a good coloring of H. In particular, a $(\Delta(H^2)+1)$-coloring of H^2, which can be greedily computed in $O(\min(|V|^2, |V|\,\Delta(H)^2)) = O(\min(|V|^2, k^2|V|))$ time, gives the desired good coloring for H.

The Algorithm. The only data structure needed by the algorithm is a list $A(u)$ of available colors which is initially set to $[k]$ for each vertex u. The algorithm is probabilistic but can be derandomized with a standard application of the method of conditional probabilities (see for example [1, 7]). The input is G and H.

1. Compute a good coloring s of H using $\ell = (k-1)^2 + 1$ colors. Let σ be a random permutation of s.
2. [Comment: the idea is to use σ as a *schedule* for the vertices in H to get their final color.]
   For $i = \sigma_1, \sigma_2 \ldots, \sigma_\ell$ do: each vertex u such that $\sigma(u) = i$ picks a final color $\chi(u)$ uniformly at random from $A(u)$. The H-neighbors of u update their lists by removing $\chi(u)$.
3. For all $u \in G \setminus H$: Each u picks a color c uniformly at random in $[k]$.

In the next two lemmas, we establish correctness of the algorithm. Afterwards the running time is analyzed.

Lemma 2. *No edge of H is monochromatic.*

Proof. Following the convention introduced in the algorithm, let $\chi(u)$ denote the final color of an H-vertex obtained in step 2 and let $\sigma(u)$ be the processing time of vertex u in step 2. Recall that σ is a random permutation of the good coloring computed in step 1.

The claim follows from a couple of observations. First, H-vertices choosing their final color $\chi(\cdot)$ in parallel form an independent set because vertices processed in parallel in step 2 have the same $\sigma(u)$. This ensures that the color choices made in parallel at iteration i are mutually compatible. Second, the way the lists are updated ensures that if u and v are neighbors in H then $\chi(u) \neq \chi(v)$. Moreover, the number of colors available to a vertex in H is always greater than the number of its H-neighbors because of the way the color lists are updated and the fact that initially $|A(u)| = k > \Delta(H)$. The claim follows.

Lemma 3. *For any edge e of $G \setminus H$, the probability that e is monochromatic is at most $2\mathcal{H}(k)/k$.*

Proof. There are three possible kinds of edges: (a) edges of H; (b) edges of $G \setminus H$ which have both endpoints in H; and (c) edges in $G \setminus H$ which have only one endpoint in H (edges in $G \setminus H$ with both endpoints not in H do not exist, otherwise they would be included in H).

Lemma 2 shows that no edge of type (a) is monochromatic. For an edge $e = (u, v)$ of type (c) we compute the probability that it is monochromatic. Let $u \in H$, $\chi(u) = c$ and $v \in G \setminus H$. Then, from step 3 of the algorithm,

$$\Pr[e \text{ monochromatic}] = \Pr[\chi(u) = c] = \frac{1}{k}.$$

To conclude the proof, let $e = (u, v)$ be an edge of type (b). Notice that the random schedule σ induces a sub-permutation among any H-vertex u and its H-neighbors and that u can occupy any position within this sub-schedule with uniform probability. We refer to u's position within this sub-permutation as the *local time* of vertex u. We need the following fact. If $A_i(u)$ and $\deg_H^i(u)$ denote, respectively, the available palette at u and the number of H-neighbors of u still uncolored at local time i then, $|A_i(u)| \geq \deg_H^i(u) + 1$. This follows from the fact that σ is a good coloring. Let $\mathrm{MT}(u, v)$ denote the maximum of the local time of vertex u and the local time of vertex v. Then,

$$\Pr[e \text{ monochromatic}] = \sum_{t \in [k]} \Pr[\chi(u) = \chi(v) | MT(u, v)) = t] \Pr[MT(u, v)) = t].$$

Assume pessimistically that when a vertex performs its color choice at local time t then all H-neighbors of u have picked different colors. Then,

$$\begin{aligned}\sum_{t \in [k]} \Pr[\chi(u) = \chi(v) | MT(u, v)) = t] \Pr[MT(u, v)) = t] &< \sum_{t \in [k]} \frac{1}{k - t + 1} \frac{2t - 1}{k^2} \\ &\leq \frac{2}{k} \sum_{t \in [k]} \frac{1}{k - t + 1}\end{aligned}$$

$$= 2\frac{\mathcal{H}(k)}{k}.$$

We now turn to the running time of the algorithm.

Step 1 can be computed as outlined above in $O(k^2|V|)$ time. Step 2 takes $O(k^2|V|)$ time and step 3 takes $O(|V|)$ time.

Step 1 can be computed in deterministic time $O(k^5|E|)$. Step 2 and step 3 can be computed in deterministic time $O(|E| + k|V|)$. Thus the algorithm runs in deterministic time $O(k^5|E| + k|V|)$.

How hard is it to find H? In the terminology of Lovasz & Plummer's book on Matching [4], computing a maximum $(k-1)$-matching is reducible to Maximum Matching in a graph with $O(|E| + |V|)$ vertices and $O(|E|^2)$ edges. Maximum Matching in a graph with n vertices and m edges can be computed in time $O(\sqrt{n} \cdot m)$ [5]. Thus H can be found in probabilistic time $O(\sqrt{|E| + |V|} \cdot |E|^2 + k^2|V|)$.

3.2 Probability $1/k$ for non-guaranteed edges

Our first result in this subsection is the following: Given a graph and a maximum 1-matching H of G one can color G such that all edges of H are bichromatic and at least half of the edges of $G \setminus H$ are bichromatic. We now give the algorithm.

Algorithm A. The algorithm is probabilistic but can be derandomized with a standard application of the method of conditional probabilities [1, 7].

1. Give each edge of H an arbitrary direction.
2. For each directed edge (u, v) of H let U receive the color 1 with probability $1/2$ and the color 2 with probability $1/2$, and let v receive the other color.

Lemma 4. *After the execution of Algorithm A no edge of H is monochromatic. Also, for any edge e of $G \setminus H$, the probability that e is monochromatic is $1/2$.*

Proof. It is trivial to see that no edge of H is monochromatic. Note that for an edge (u, v) of $G \setminus H$ the colors of u and v are independent. Hence, the probability that an edge of $G \setminus H$ is monochromatic is $1/2$.

Algorithm A runs in $O(|V|)$ probabilistic time and the derandomized variation of it runs in $O(|V| + |E|)$ deterministic time.

The above result can be generalized to the following. Given a graph G and a maximum $((\log k) - 1)$-matching H of G, G can be colored such that no edge of H is monochromatic and at most a fraction $1/k$ of the edges of $G \setminus H$ becomes monochromatic. We now describe the algorithm. In the algorithm, each vertex v of G is colored with a vector $\chi(v) \in \{0, 1\}^{\log k}$.

Algorithm B. The algorithm is probabilistic but can be derandomized with a standard application of the method of conditional probabilities [1, 7].

1. Give each edge of H an arbitrary direction.
2. Edge-color H using the colors $[\log k]$
3. For each directed edge (u, v) of H with color i, set $x(u)_i$ to 0 with probability $1/2$ and to 1 with probability $1/2$, and let $x(v)_i$ be $x(u)_i \oplus 1$.

4. For each color i that do not occur at a vertex v set, $x(v)_i$ to 0 with probability 1/2 and to 1 with probability 1/2.

Lemma 5. *After the execution of Algorithm B no edge of H is monochromatic. Also, for any edge e of $G \setminus H$, the probability that e is monochromatic is $1/k$.*

Proof. It is trivial to see that no edge of H is monochromatic. Note that for an edge (u, v) of $G \setminus H$ the colors of u and v are chosen independently and uniformly from among k colors. Hence, the probability that an edge of $G \setminus H$ is monochromatic is $1/k$.

Algorithm B runs in $O(|V| \log k)$ probabilistic time and the derandomized variation of it runs in $O((|V| + |E|) \log k)$ deterministic time.

3.3 Probability $O(1/k)$ for non-guaranteed edges

In the next section, we will show that when H is a d-inductive graph, we can obtain probability $1/(k-d)$ that an edge of $G \setminus H$ is monochromatic. However, we already now note that this implies the following.

Lemma 6. *For each $0 < c < 1$ when H is a maximum ck-matching in G we can in polynomial time k-color G such that (1) no edge of H is monochromatic (2) the probability that an edge of $G \setminus H$ is monochromatic is $1/((1-c)k)$.*

4 Inductive graphs

In this section, we show that when H is a d-inductive subgraph of G then G can be k-colored such that (1) no edge of H is monochromatic (2) the probability that an edge of $G \setminus H$ is monochromatic is $1/(k-d)$. We will then apply this result to the case when H is a union of t trees by showing that any such union is $2t-1$-inductive.

4.1 Algorithm for d-inductive subgraphs

Algorithm. The algorithm is probabilistic but can be derandomized with a standard application of the method of conditional probabilities [1, 7]. We are given G and a d-inductive subgraph H of G where $d < k$. We shall color G with k colors.

1. Delete a vertex v of degree d in H from H.
2. Color $H \setminus \{v\}$ recursively.
3. Color v with a color chosen uniformly from the colors not present in the neighborhood of v in H.

Lemma 7. *After the execution of the algorithm no edge of H is monochromatic. Also the probability that an edge of $G \setminus H$ is monochromatic is $1/(k-d)$.*

Proof. It is trivial to see that no edge of H is monochromatic. Note that for an edge (u, v) of $G \setminus H$ the colors of u and v are chosen independently and uniformly from among $k-d$ colors. Hence, the probability that an edge of $G \setminus H$ is monochromatic is $1/(k-d)$.

The algorithm runs in $O(k|V|)$ probabilistic time. Its derandomized variation runs in $O(|E| + k|V|))$ deterministic time.

4.2 A union of t trees

In this subsection, we show that a union of t trees is $2t-1$ inductive. Hence, when H is such a union we can k-color G such that (1) no edge of H is monochromatic (2) the probability that an edge of $G \setminus H$ is monochromatic is $1/(k-2t+1)$. We also show that such an H is easy to find.

To chose a heavy union of t trees we will generate a sequence of graphs $G_1, G_2, \ldots, G_t, G_{t+1}$, a sequence of subgraphs $H_1, H_2, \ldots, H_t$, and a sequence of MSTs $T_1, T_2, \ldots, T_t$ such that: (a) T_i is a MST in G_i; (b) $G_1 = G$ and $G_{i+1} = G_i \setminus T_i$; and $H_1 = T_1$ and $H_{i+1} = H_i \cup T_{i+1}$. We will then let $H = H_t$.

Lemma 8. *A union of t trees is $2t$-colorable.*

Proof. Let H be a union of t forests. We make two observations. First, if v is a vertex of H then $H \setminus \{v\}$ is a union of t forests. Second, a union of t forests contains a vertex of degree $2t-1$. The latter holds since if the union contains n vertices then the degree sum in the union is at most $2t(n-1)$ and this implies that one vertex has degree at most $\lfloor (2t(n-1)/n) \rfloor = 2t-1$. ¿From these two observations follows that H is $2t-1$ inductive; and this clearly implies that it is $2t$-colorable.

The Algorithm.

1. $G_1 := G$; $H_0 := \emptyset$
2. **for** $i := 1$ **to** t **do**
   Let T_i be a maximum spanning tree in G_i computed using Prim's algorithm.
   $H_i := H_{i-1} \cup T_i$
   $G_{i+1} := G_i \setminus T_i$
   $H := H_t$

How hard is it to find H? One maximum spanning tree can be constructed using Prim's algorithm in $O(|E| + |V| \log |V|)$ time. The set operations take $O(|E|)$ time. It follows that we can find H in $O(t(|E| + |V| \log |V|))$ deterministic time.

References

1. N. Alon, J. Spencer and P. Erdös, The Probabilistic Method, John Wiley & Sons, 1992.

2. A. Frieze and M. Jerrum, Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In *Proc. of 4th Int. Conf. on Integer Prog. and Combinatorial Optimization*, Lecture Notes in Comput. Sci. 920, pages 1–13. Springer-Verlag, 1995.
3. V. Kann, S. Khanna, J. Lagergren and A. Panconesi, On the hardness of approximating MAX k-CUT and its dual. KTH Tech. report TRITA-NA-9505, 1995. Submitted.
4. L. Lovasz and M. D. Plummer, Matching Theory, Annals of Discrete Mathematics 29, North Holland, 1986.
5. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In Proc. 21st Ann. Symp. on Foundations of Computer Science, pages 17-27, IEEE, 1980.
6. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. J. Comput. System Sci., 43:425–440, 1991.
7. P. Raghavan, Lecture notes on randomized algorithms, Tech. report RC 15340 (#68237), IBM T. J. Watson Research Center, January 1990.