Latex Template

Víctor Alcázar

June 1, 2017

Contents

1 Statement

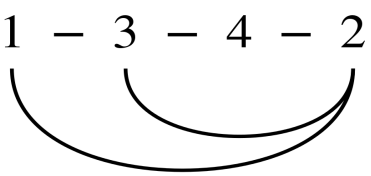
We've been given the following problem to talk about.

Minimum cut linear arrangement
Given a graph $G = (V, E)$, compute a one-to-one function $f : V \to [1..|V|]$ so that the maximum number of cut edges in any integer point is minimised, i.e.

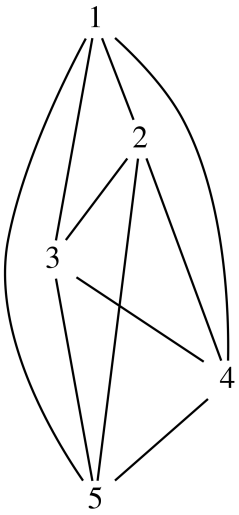
$$\max_{i \in 1..|V|} |\{\{u, v\} \in E : f(u) \leq i < f(v)\}|$$

2 Example Instances

In order to visualize the problem and attack it using more intuition, we must consider that the cuts at integer points can be seen easily in the following manner. Given a graph G and an ordering function f, we draw the nodes in an horizontal line, in the order given by f. What we are trying to minimize is the size maximum cut among the S_i's , where $S_i = \{v \in V : f(v) \leq i\}$, for $i = 1, 2, \ldots, n-1$. But on the graph drawing mentioned above, the size of every S_i is the number of edges existing in the space between the vertical lines passing by nodes i and i+1. To give an example, in the following linear arrangement we easily deduce that the minimum S_i size is 3:

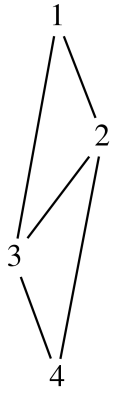


TODO: draw another example
The problem instances are Graphs G. As an example, we take $G = K_5$:

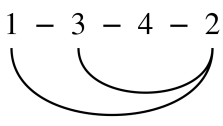


Here, every node is connected to every one of the rest. Therefore, every node is identical to every one else. In this case, every assignment of numbers $1 \dots n$ to the nodes yields the same graph. Therefore to solve this instance, we consider a random labeling of our nodes, and we compute the cuts at all integer points between 1 and n.

Another instance, could be this graph.



Here, the assignment matters. In order to attack the problem in a more intuitive manner, it would be more suitable to have visualize the problem as follows: Given graph G, create a drawing of G such that the nodes are placed on a straight horizontal line. The objective is to minimize the maximum number of edges that exist in the space between two consecutive nodes. For example, the graph below is a visualization of the assignment $f(1) = 1, f(2) = 4, f(3) = 2, f(4) = 3$, which gives us $max(S_i) = 3$.



By the way, the above assignment gives us the best possible results, w.r.t. the requirements of the problems. That is, there is no assingment that gives us a lower maximum cutwidth. **explain better here**

3 Decisional Problem

Given a graph $G = (V, E)$ and $k \in N$, say if there exists one ordering of the vertexs so that the maximum number of cut edges in any integer point is less than or equal k, i.e.

$$\exists f : V \to [1..|V|] | \max_{i \in 1..|V|} | \{\{u, v\} \in E : f(u) \leq i < f(v)\} | \leq k$$

4 Parametrized Problems

Using this problem, and combining it with a computable function $\kappa : \Sigma^* \to N$ we can formulate a Parametrized version of this problem.

TODO: Add Natural numbers symbol

An example:

Input	Graph G and tw = treewidth(G)
Paremeters	tw
Question	What is the Min-Cut Linear Arrangement of G?

Given that we find an expression Monadic Second Order Logic, the above can be solved in $O(n * f(tw))$ /citetrans.

5 A $(log^2(n))$ Approximation

There exists an algorithm that runs in subexponential time and achieves an $(log^2(n))$ approximation of the min.cut linear arrangement problem.

Source: [**?**]

TODO: examine explain how the time is subexponential TODO: explain the proof below in a better way (look at Vazirani-Approximation algorithms)

Problem 21.27 (Minimum b-balanced cut)

Given an undirected graph G = (V, E) with nonnegative edge costs and a rational b, 0 ¡ b =¡ 1/2, find a minimum capacity cut (S, S) such that b * n ¡= —S— ¡ (1 b) * n. A b-balanced cut for b = 1/2 is called a bisection cut, and the problem of finding a minimum capacity such cut is called the minimum bisection problem. We will use Theorem 21.24 to obtain a pseudo-approximation algorithm for Problem 21.27 – we will find a (1/3)-balanced cut whose capacity is within an O(log n) factor of the capacity of a minimum bisection cut (see the notes in Section 21.8 for a true approximation algorithm). For V V , let G V denote the subgraph of G induced by V . The algorithm is: Initialize U and V V . Until —U — n/3, find a minimum expansion set in G V , say W , then set U U W and V V W . Finally, let S U , and output the cut (S, V S).

Problem 21.29 (Minimum cut linear arrangement) Given an undi- rected graph G = (V, E) with nonnegative edge costs, for a numbering of its vertices from 1 to n, define S i to be the set of vertices numbered at most i, for 1 ¡= i ¡= n 1; this defines n 1 cuts. The problem is to find a numbering that minimizes the capacity of the largest of these n1 cuts, i.e., it minimizes maxc(S i)— 1 ¡= i ¡= (n 1). Using the pseudo-approximation algorithm obtained above for the (1/3)-balanced cut problem, we will obtain a true O(log 2 n) factor approximation algorithm for this problem. A key observation is that in any arrangement, S n/2 is a bisection cut, and thus the capacity of a minimum bisection cut in G, say , is a lower bound on the optimal arrangement. The reason we get a true approximation algorithm is that the (1/3)-balanced cut algorithm compares the cut found to . The algorithm is recursive: find a (1/3)-balanced cut in G V , say (S, S), and recursively find a numbering of S in G S using numbers from 1 to —S— and a numbering of S in G S using numbers from —S— + 1 to n. Of course, the recursion ends when the set is a singleton, in which case the prescribed number is assigned to this vertex. Claim 21.30 The algorithm given above achieves an O(log 2 n) factor for the minimum cut linear arrangement problem.

6 Various results

In this section we will present a number of results regarding our problem.

Yanakakis presents an algorithm that finds a min-cut linear arrangement of a tree in O(n log n) time [**?**]

The MINCUT problem for genera1 graphs is NP-complete [11,13]. The restriction of the problem to trees has been open for some time. Lengauer described an approximation algorithm in [16] that produces a layout with cutwidth at most twice the optimal. He also determined exactly the cutwidth of complete k-ary trees. F. R. K. Chung studied the properties of optimal layouts [2]. M. Chung et al. present in [5] an algorithm that solves the MINCUT problem on trees in time O(n(log n)d-2) where d is the maximum degree of the tree. Thus, their algorithm works in polynomial time for bounded degree trees, but exponential time in genexal. Dolev and Trickey [7] study the MINCUT problem on trees and give an O(n log n) algorithm for a planar version of it, where no edge crossings are allowed if the tree is drawn as in Figure 1 with all the edges above the line where the nodes lie.

TODO: explain well this part of Yanakakis' paper. Its uploaded in the repo TODO: obviously, add the bibliography

References

[1] Maria Serna, *Class transparencies*, Fall 2017

Vijay Vazirani *Approximation Algorithms*, 2001, Springer

Mihalis Yannakakis *A Polynomial Algorithm for the Min-Cut Linear Arrangement of Trees* 1985, ATT Bell Laboratories, Murray Hill, New Jersey