Lecture # 2: The Partitioning Approach

1 Introduction

The focus of this lecture is on a simple technique: partitioning the elements of the input into pieces that are easy to solve. This often gives good results, especially for weighted problems. We shall also derive a few exact algorithms for solving the "easily-solvable" subproblems.

We focus on the WEIGHTED INDEPENDENT SET (WIS) problem, which is representative for various maximum subgraph problems.

1.1 Partitioning approach

The following proposition represents the elementary version of the partitioning idea. Recall that a *partition* of a set V is a collection of subsets $V_1, \ldots, V_t$ such that $\cup_{i=1}^{t} = V$ and $V_i \cap V_j = \emptyset$ for all $i \neq j$.

Proposition 1.1 *Let $G = (V, E)$ be an instance of* WIS. *Suppose we can partition V into t subsets $V_1, V_2, \ldots, V_t$ such that* WIS *is polynomial solvable on each induced subgraph $G[V_i]$. Then, the largest of these t solutions ($WIS(G[V_1])$, $WIS(G[V_2])$, $\ldots$, $WIS(G[V_t])$) is a t-approximation to $WIS(G)$.*

This is easy to argue. Consider the optimal solution OPT, and restrict it to V_i, obtaining a set $OPT \cap V_i$. This is an independent set contained in V_i and thus a feasible solution of WIS on $G[V_i]$. Hence, the weight of $OPT \cap V_i$ is at most $WIS(G[V_i])$. Therefore, we have

$$\begin{aligned} w(OPT) &= w(OPT \cap V_1) + w(OPT \cap V_2) + \cdots + w(OPT \cap V_t) \\ &\leq WIS(G[V_1]) + WIS(G[V_2]) + \cdots + WIS(G[V_t]) \\ &\leq t \cdot (\max_i WIS(V_i)) \\ &= t \cdot w(ALG). \end{aligned}$$

As usual, ALG denotes the solution output by the algorithm under study. Therefore, the approximation ratio is at most t.

We are ready to look the first application of the principle.

WIS in general graphs

Theorem 1.2 WIS *can be approximated within $n/\log n$ factor (in polynomial time).*

Proof. Given weighted graph $G = (V, E)$, partition V *arbitrarily* into $t = n/\log n$ sets, each containing $\log n$ vertices. For simplicity of presentation, we ignore the rounding.

We solve each subgraph $G[V_i]$ exhaustively. Namely, we consider every subset of V_i, test if is independent, and output the maximum weight independent found. Since V_i contains $x = \log n$ nodes, it contains $2^x = n$ subsets. Testing each is done in time $\binom{x}{2} = O(\log^2 n)$. There are $n/\log n$ subgraphs. Hence, the total complexity is on the order of

$$n/\log n \cdot 2^{\log n} \cdot \log^2 n = n^2 \log n.$$

Exercise: Show that for *any* integer c, WIS can be approximated within $n/(c \log n)$ factor in polynomial time.

2 Partitioning lemma of Lovász

The following lemma (and its generalization) is in one of Lovász's very first papers.

Lemma 2.1 (Lovász [?]) *Let $G = (V, E)$ be graph of maximum degree Δ, and let k be a positive integer. Then, V can be partitioned into $t = \lceil (\Delta + 1)/(k + 1) \rceil$ subsets $V_1, \ldots, V_t$, such that $\Delta(G[V_i]) \leq k$ for $i = 1, 2, \ldots t$.*

The somewhat surprising result of this is that the sum of the maximum degrees of the subgraphs adds up to only about $k/(k + 1)$-fraction of the original maximum degree.
Proof. We give a *local search* algorithm that outputs a partition satisfying the lemma.

> Start with an arbitrary partition. If there is a vertex v of degree more than k in the current subgraph, move it to a subgraph where it has k or fewer neighbors. Repeat the above operation as often as needed.

First, observe that for any vertex, there is always some subgraph in which it has at most k neighbors, since otherwise it would have at least $t(k + 1) = \lceil (\Delta + 1)/(k + 1) \rceil \cdot (k + 1) \geq \Delta + 1$ neighbors. Second, we need to show that the algorithm converges on a solution, and bound the number of iterations.

The tool we use to show convergence is to use a *potential function*. For a partitioning P, let Φ_P denote the number of *cross edges* in the graph according to that partition. Namely,

$$\Phi_P = |\{uv \in E(G) : \exists i, j, i \neq j \text{ s.t. } u \in V_i \text{ and } v \in V_j\}|.$$

Consider a particular iteration during the course of the algorithm. Let v be the vertex being moved from part V_i to V_j; let P be the partition before the iteration, and P' after the iteration. We examine the effect of the operation on the potential function. Clearly, the only edges affected by the moves are those incident on v. Further, only edges from v to vertices in either V_i or V_j are affected. In fact, the form edges are edges that were not cross edges in P but come cross edges in P', and the vice versa for the latter Thus,

$$\Phi_{P'} - \Phi_P = |\{u \in G_j : uv \in E\}| - |\{u \in G_i : uv \in E\}| \geq (k + 1) - k = 1.$$

Hence, the potential function increases by at least one unit in each step. Initially, it is non-zero, and in the end it is at most $|E| \leq \Delta n$. Hence, the time complexity is polynomial. ∎

This lemma has several elegant applications to the approximation of induced subgraph and vertex partitioning problems. A property of graphs is said to be *hereditary* if, whenever it holds for a graph it also holds for any induced subgraph.

Theorem 2.2 WIS *can be approximated within a $\lceil (\Delta + 1)/3 \rceil$ factor.*

Proof. Set $k = 2$, and apply Lovász's lemma. We obtain $t = \lceil \frac{\Delta+1}{3} \rceil$ subgraphs that are of maximum degree at most 2. Such graphs consist of disjoint paths and cycles. It is an (important!) exercise to show that WIS can be solved optimally on such graphs in linear time using dynamic programming. By the partitioning proposition, we thus obtain a t-approximation. ∎

We can obtain slight improvements to this, using some previous results. This allows us to illustrate a slight variation in the partitioning approach.

The previous best approximation for the weighted independent set problem is $\frac{2}{\Delta}$ due to Hochbaum [**?**]. We can use her approximation for $\Delta = 3$ to improve our ratio when Δ is a multiple of 3.

Theorem 2.3 WIS *can be approximated within* $(\Delta + 2)/3$.

Proof. When $\Delta \bmod 3$ is congruent to 1 or 2, the claim follows from Theorem 2.2. When $\Delta \bmod 3 = 0$, partition the vertices into $\frac{\Delta}{3}$ classes, where all but possibly the last have maximum degree 2. Find a 3/2-approximate weighted independent set in the last class by Hochbaum's result, compute optimal solutions of the other classes, and let output the largest of all of these. Let W denote the weight of our solution.

The weight of the optimal solution is at most the sum of the weights of the optimal solutions on the $\frac{\Delta}{3}$ subgraphs. The weight of the optimal solution of the last subgraph is at most $\frac{3}{2}W$, but at most W for the other subgraph. Hence, the weight of the optimal solution is at most $[\frac{3}{2} + (\frac{\Delta}{3} - 1)]W = [\frac{\Delta}{3} + \frac{1}{2}]W$. ∎

Exercise: Give a linear-time algorithm for WIS on cycle graphs (regular graphs of degree 2).

3 WIS on planar graphs

We now consider a technique of Baker [**?**] for approximating WIS in planar graphs.

A *planar graph* is a graph that can be embedded in the plane so that no edges cross. A planar graph along with a planar embedding is called a *plane graph*. The embedding partitions the plane into *faces*, of which one is the *infinite face*. Figure 3 shows a plane graph. In it, all but the infinite face contain three vertices.

Baker used the following *decomposition* of the planar graph. Let V_1 be the vertices on the *convex hull* of the plane graph; that means, all the vertices on the infinite face of the graph. This subgraph has the property of being an *outerplanar graph*: No vertex is closed inside a circle. See Figure 1.

Delete V_1 and incident edges from the graph, obtaining $V' = V - V_1$. Repeat the "peeling-off" operation on V', obtaining a partition of V into *layers* $V_1, V_2, \ldots, V_t$. It has the following useful *separation property*:

V_i and V_j are disconnected, whenever $i \geq j + 2$.

In other words, vertices in V_i are adjacent only to vertices in V_{i-1}, V_i and V_{i+1}.

This also implies that the graph induced by the vertices in odd-numbered layers $V_o = V_1 \cup V_3 \cup V_5 \cdots$ is also outerplanar. Same with $V_e = V_2 \cup V_4 \cup V_6 \cdots$. We state this observation. (Recall that by "partitioning a graph into subgraphs" we mean partitioning the vertex set into subsets inducing the particular subgraphs.)

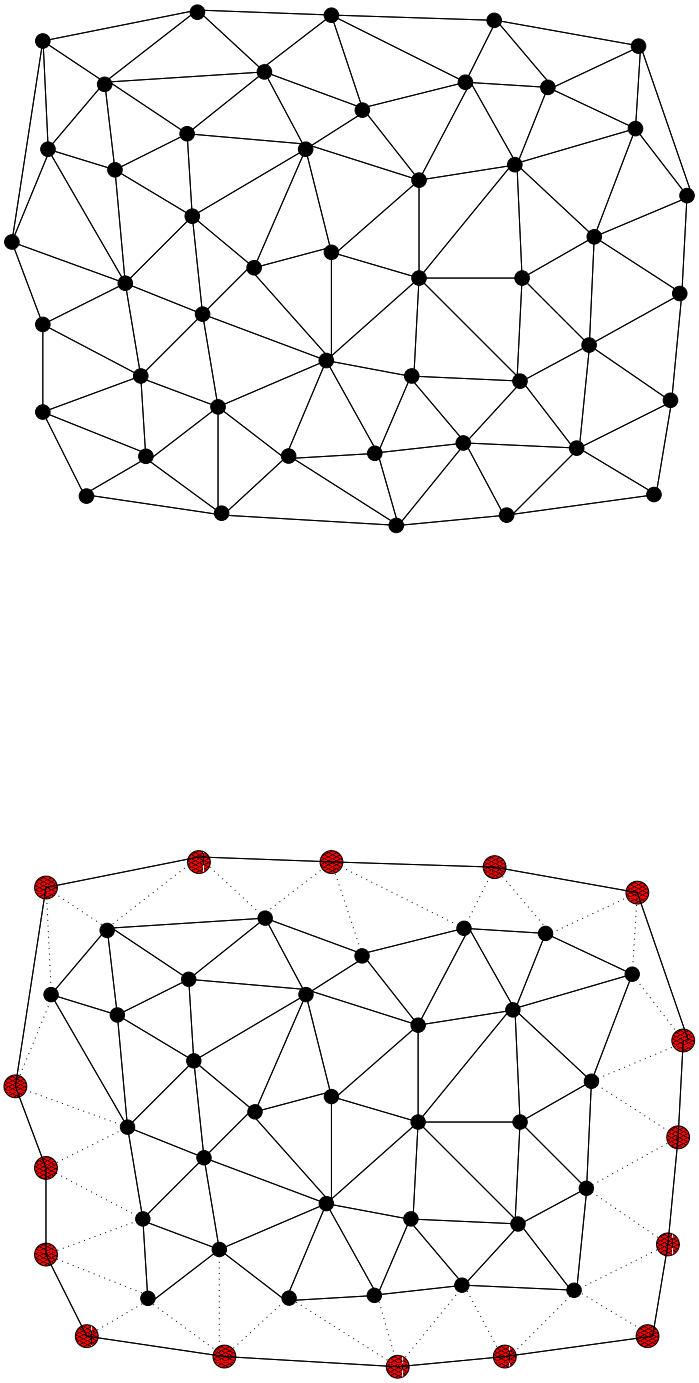

Figure 1: First layer “peeled off”

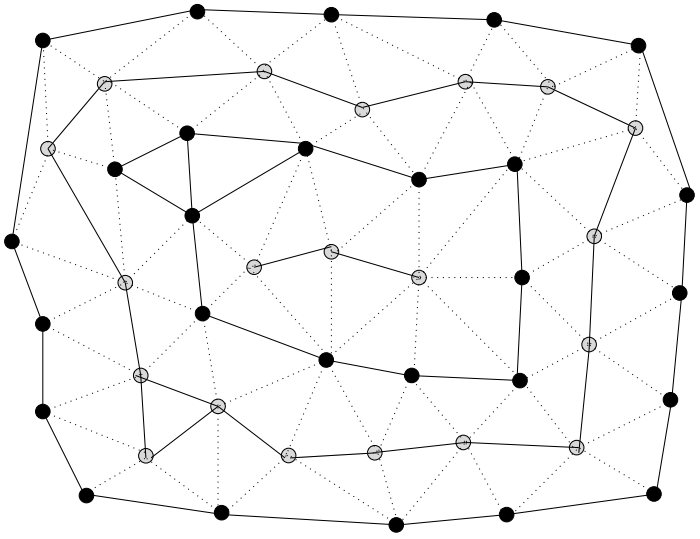

Figure 2: Partition into layers

Observation 3.1 *A planar graph can be partitioned into two outerplanar graphs.*

We shall soon prove the following result.

Lemma 3.2 WIS *can be solved optimally on outerplanar graphs in linear time.*

We are now ready for the first approximation result, which follows immediately from Observation 3.1, Lemma 3.2, and the Partitioning property.

Theorem 3.3 WIS *on planar graphs can be approximated within a factor 2 in linear time.*

3.1 WIS on outerplanar graphs

We now consider how to apply dynamic programming to solve WIS optimally in linear time on an outerplanar graph. We do so by showing how outerplanar graphs belong to a more general class of graphs, known as *partial k-trees* (here $k = 3$), for which a lot of work has been done on efficient algorithms.

A *tree decomposition* of a graph G is a tree $\mathcal{T}$, and a labeling $x : V(\mathcal{T}) \mapsto 2^{V(G)}$ of the vertices of the tree by sets of vertices in G, satisfying the following two properties:

1. Consider a vertex v of G. The vertices of $\mathcal{T}$ whose label contains v forms a connected subgraph (subtree) of $\mathcal{T}$.

2. For any edge uv in G, some label of $\mathcal{T}$ contains both u and v.

The *treewidth* of a tree-decomposition is the maximum cardinality of any label of the tree.

Let us consider an example. Figure 3 contains an outerplanar graph G. Vertices are labeled 1 through 9, and all edges are "inside" the circle.

A tree decomposition of G is given in Figure 4.

One informal algorithm for generating a tree decomposition of an outerplanar graph might be as follows. Suppose the vertices lie on a path, and are labeled accordingly.

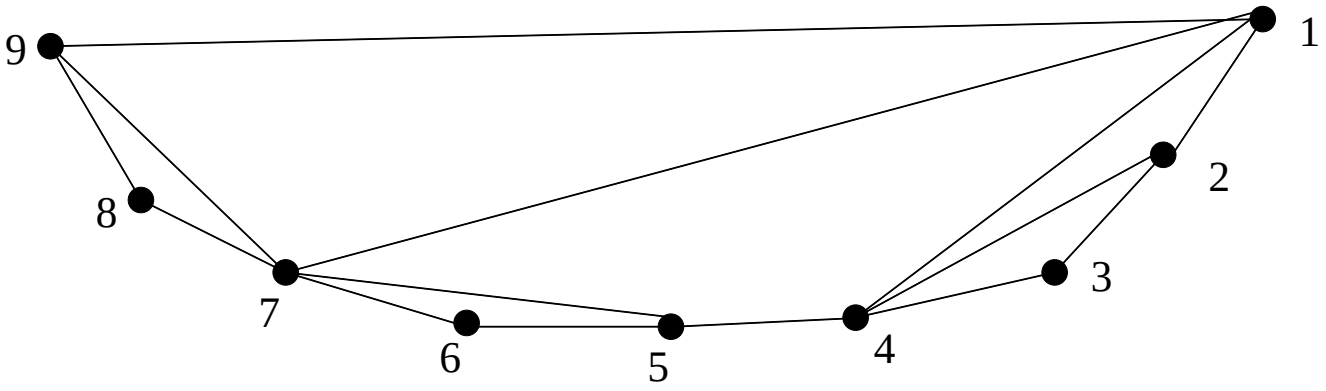

Figure 3: An outerplanar graph

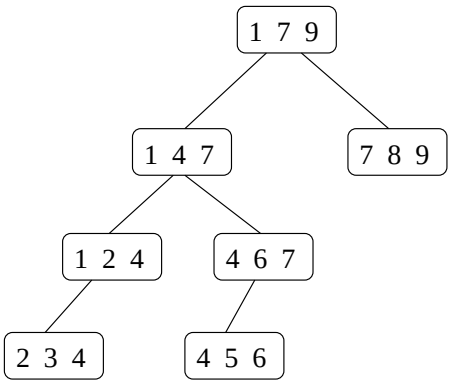

Figure 4: A tree decomposition

Treedecomp(G)
Let a (b) be the smallest (largest) numbered node.
Let c be the largest neighbor of a.
Let G_1 be the subgraph of G induced by vertices $a...c$ if $c \geq a+2$
Let G_2 be the subgraph of G induced by vertices $c...b$.
Let abc be the label of the root r of $\mathcal{T}$.
The left subtree of r is given by TreeDecomp(G_1), if G_1 contains 3 or more nodes, but is empty otherwise.
Similarly, the right subtree of r is given by TreeDecomp(G_2).

Observe that abc forms a *cut* in G. Thus, nodes that appear in left subtree of $\mathcal{T}$ will not appear in the right subtree of $\mathcal{T}$, and vice versa. It is also easy to verify that all edges appear as pairs in some label of the tree. It then follows that we obtain a valid tree-decomposition. Clearly, the treewidth of the decomposition is 3.

WIS **on treewidth-k graphs** We now give a dynamic programming algorithm for finding the *weight* of a maximum WIS of a graph with a given tree decomposition. The complexity of the algorithm is $O(2^k n)$. To compute the independent set matching the optimum value, we can retrace the steps of the algorithm backwards, as is common with dynamic programming.

For a treenode u, let V_u be the set of vertices of $V(G)$ appearing in the subtree rooted at u. Associated with u is the subproblem A_u: Compute the WIS of $G[V_u]$. We compute A_u for each treenode; A_r then gives $WIS(G)$, where r is the root of $\mathcal{T}$.

We maintain a table of size 2^k for each treenode u. Namely, there are 2^k possibilities for including vertices of the treenode label in the optimum WIS solution of A_u. For each possibility, we store the weight of the optimum WIS of $G[V_u]$ *under the constraint* that the particular nodes are contained/not contained in the solution.

To compute a particular value in the table at u, we examine all values in the tables at the left and right child of u, and keep the possibility that gives maximum return.

Let us consider a hypothetical example of a 2-tree, with a root node labeled ab with childrens labeled ac and bd:

a	b	wt
0	0	$20(12+8)$
0	1	$21(12+9)$
1	0	$21(13+8)$
1	1	$18(13+5)$

a	c	wt
0	0	11
0	1	12
1	0	7
1	1	13

b	d	wt
0	0	2
0	1	8
1	0	9
1	1	5

The complexity of the approach is at most $O(2 \cdot 2^k)$ for each entry of the updated table, for a complexity per table of $O(4^k)$. Hence, total of $O(4^k n)$ for the whole graph. Some optimizations are possible in practice.

Exercise: Try to reduce the complexity of WIS on treewidth-k graphs to $O(2^k kn)$.

3.2 Improved approximation

An improved approximation for planar graphs can be achieved by giving an optimal algorithm for a class of graphs that subsumes outerplanar graphs, and intuitively covers a larger portion of the plane.

A *k-outerplanar graph* is a union of up to k layers of Baker's decomposition. Namely, the graph induced by $V_1 \cup V_2 \cup V_3 \cup \cdots \cup V_k$ is k-outerplanar. We shall not prove the following result

of WIS, which uses the same ideas as for outerplanar graphs. It can also be shown by using a result of Bodlaender that k-outerplanar graphs have treewidth at most $3k-1$.

Lemma 3.4 (Baker) WIS *can be computed optimally on k-outerplanar graphs in time $O(8^k n)$.*

Notice that by the separation property, more generally, the graph induced by $V - (V_{k+1} \cup V_{2k+1} \cdots)$ is also k-outerplanar. Roughly speaking, it covers all but a $1/k$-fraction of the graph. This suggests the following decomposition.

Let k be fixed natural number. For $i = 1, 2, \ldots, k$, let

$$U_i = V - \cup_{j=0} V_{jk+i}.$$

Note that each U_i is k-outerplanar. We now use a more general version of the Partitioning property.

Proposition 3.5 *Let $G = (V, E)$ be a vertex-weighted graph, and let $U_1, \ldots, U_k$ be subsets of V such that each vertex in V appears at least t times (i.e. in t different U_i). Then, if $ALG = \max_i WIS(G[U_i])$, then $OPT/ALG \leq k/t$.*

Proof. As before,

$$\begin{aligned} w(ALG) &= \max_i WIS(U_i) \\ &\geq \frac{1}{k}(WIS(G[U_1]) + WIS(G[U_2]) + \cdots + WIS(G[U_k])) \\ &\geq \frac{1}{k}(w(OPT \cap U_1) + w(OPT \cap U_2) + \cdots + w(OPT \cap U_k)) \end{aligned}$$

However, since each vertex appear t times in the U_i's, we have that

$$w(OPT) \leq \frac{1}{t}(w(OPT \cap U_1) + w(OPT \cap U_2) + \cdots + w(OPT \cap U_k)).$$

∎

In particular, in our setup, $t = k-1$. Thus, we obtain the following improved approximation.

Theorem 3.6 WIS *on planar graphs can be approximated within $k/(k-1)$ in time $O(8^k n)$.*

Corollary 3.7 WIS *on planar graphs can be approximated within $1 + 1/\log n$ in polynomial time.*