University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 19: Deterministic Minimum Cuts via Expander Decomposition

November 16, 2021

Instructor: Thatchaphol Saranurak

Scribe: Chaitanya Nalam

We have seen from previous lecture that we can use expander decomposition to get spanners and cut-sparsifiers which reduce the number of edges in the graph but preserving the important information from the graph like shortest distance between vertices (or) cut edges of any subset of vertices in graph etc., although at a loss in factor of approximation.

Today, we will see how to use expander decomposition help us solve some problem exactly.

The spanners and cut-sparsifiers seen in previous class reduce the number of edges in the graph maintaining the same vertex set which comes under a broad regime of Edge Sparsification. There is a similar notion with respect to vertics called Vertex Sparsification where we preserve some properties of the original graph by reducing the number of vertices. In this lecture we will see how we can preserve every min-cut of the graph exactly in a smaller graph with less number of vertices called **Exact Mincut Sparsifier**.

1 Introduction and Problem

One of the very well studied problems in graph theory is the problem of "Max-flow min-cut" which is as follows:

- Input: a graph $G = (V, E)$ with n vertices and m edges
- Output: a set $S \neq \emptyset$ and $S \subset V$ where $|E(S, V \setminus S)|$ is minimized

The set S is called min-cut of the graph G and $|E(S, V \setminus S)|$ is the size of the minimum cut denoted by $\lambda(G)$ also called edge connectivity of the graph G.

In this lecture we restrict ourselves to the case where G is unweighted and undirected. We only the find the value of $\lambda(G)$ however we can also find the min-cut easily. Below table summarizes the state of the art for calculating edge connectivity $\lambda(G)$ of the graph G.

Reference	Time	Det/Rand	Note
Kawarabayashi Thorup [STOC'15]	$m \log^{12} n$	Det	Complicated (Page Rank)
Henzinger Rao Wang [SODA'17]	$m \log^2 n \log\log n$	Det	Complicated (Local Flow)
Ghaffari Nowicki Thorup [SODA'20]	$m \log n, m + n \log^3 n$	Rand	Simple
This lecture	$m^{1+o(1)}$	Det	Simple, conceptually

All the algorithms listed in above table follow the same framework of Mincut-preserving contractions where vertices are contracted which fall on completely one side of any min-cut, so not contracting any cut edges of mincut thus preserving every mincut of the graph.

2 Overview and Runtime analysis

Given an input graph G we create a contracted graph G' from G such that $|E(G')| = O(m/\delta)$ and $|V(G')| = O(n/\delta)$ where δ is the minimum degree of the graph G with the guarantee that every non-trivial mincut of the graph G is preserved in G'. Non-trivial min-cuts of the graph are cuts such that both $|S|, |V \setminus S| \geq 2$ (i.e. every cut other than the singleton cuts are non-trivial).

From this we can see that

Lemma 2.1. $\lambda(G) = \min(\lambda(G'), \delta)$

Proof. The minimum cut in graph G can be either trivial cut or non-trivial cut. If it is trivial cut then $\lambda(G) = \delta$.

If it is non-trivial minimum cut then according to the guarantee it is preserved in G' thus we have $\lambda(G) \geq \lambda(G')$.

Every cut in G' is mapped to a cut in G hence minimum cut in G' cannot be smaller than minimum cut in G so $\lambda(G') \geq \lambda(G)$ so $\lambda(G) = \lambda(G')$. □

From the work of [Gab95] we have the following theorem.

Theorem 2.2. *Edge connectivity $\lambda(G)$ of a graph can be computed in $O(\lambda m \log(n^2/m))$.*

Hence given graph G' we can compute $\lambda(G')$ in $\tilde{O}(\lambda(G')|E(G')|)$ where we know that $\lambda(G') = \lambda(G)$ is at most $\delta(G)$ (the min degree), hence we can compute $\lambda(G')$ in $\tilde{O}(m/\delta(G) \cdot \delta(G)) = \tilde{O}(m)$. So we are done once we compute G'.

We use *Expander decomposition* from the work of [CGL+19] to construct G'. Formally we use the following theorem from their work.

Theorem 2.3. *Given a graph $G = (V, E)$ and expansion parameter α we can return a partition $\mathcal{P} = \{X_1, X_2, \cdots X_k\}$ of vertices V in $O(m\gamma)$ time where $\gamma = n^{o(1)}$, such that each induced sub graph on X_i i.e. $G[X_i]$ is an $\alpha-$expansion expander and inter partition edges in the graph are small i.e. $O(\alpha n\gamma)$.*

Note: This variant of decomposition is slightly different from the one given in the paper but it follows from the work with slight modifications.

Using expander decomposition stated above we construct G' in $O(m\gamma)$ time with number of edges $|E(G')| = O(m\gamma/\delta)$ where $\gamma = n^{o(1)}$. Then using [Gabow] we get $\lambda(G)$ in time $\tilde{O}(m\gamma)$ instead of $\tilde{O}(m)$ as stated.

3 Computing G'

As we have stated above we will use expander decomposition to construct G' from $G = (V, E)$. Let C be a non-trivial min-cut of graph G and we will use λ to denote $\lambda(G)$ from now.

Computing G' can be summarised in the following simple steps.

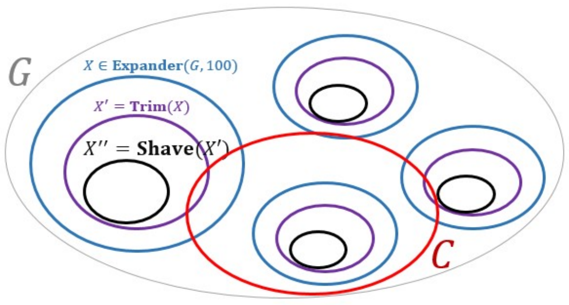

Figure 1: Computing G' (overview)

1. Compute expander decomposition of G with parameter $\alpha = 100$ giving us the partition $\mathcal{P}$ of vertices V.
$$\mathcal{P} = Expander(G, 100)$$
2. Apply a routine called $Trim$ to each of the partition $X \in \mathcal{P}$ in expander decomposition to give a smaller vertex set X'.
$$\mathcal{P}' = \{X' = Trim(X) | X \in \mathcal{P}\}$$
3. Apply a routine called $Shave$ to each set of vertices $X' \in \mathcal{P}'$ to give a still smaller vertex set X''.
$$\mathcal{P}'' = \{X'' = Shave(X') | X' \in \mathcal{P}'\}$$
4. Finally contract each of the vertex sets $X'' \in \mathcal{P}''$ to obtain graph G'.
$$G' = G/\mathcal{P}''$$
(contract each $X'' \in \mathcal{P}''$).

From above process to ensure that G' preserves all min-cuts of the graph G, it should be that for every min-cut C of graph G we should have for all vertex sets $X'' \in \mathcal{P}''$ should either be totally contained in min-cut (or) have no overlap with min-cut so that contracting these sets will not affect any min-cut of G thus preserving all of them.

Key Property:
$$\text{For all } X'' \in \mathcal{P}'', X'' \subseteq C \text{ (or) } X'' \cap C = \emptyset$$

In the remainder of this section we will define and show why applying such routines will help us satisfy the key property required above.

3.1 Overlap with Expander is small

Since, first step is to use the expander decomposition let us formally define the expander decomposition.

Expander Decomposition:

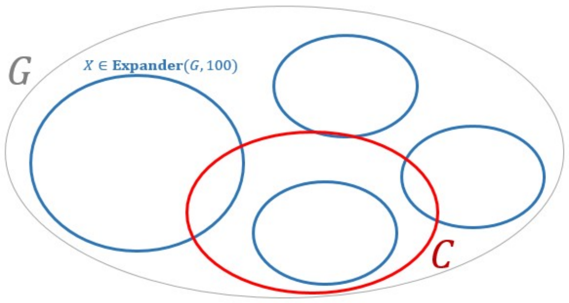

Figure 2: Expander decomposition of G

- Input: $G = (V, E)$, expansion parameter α
- Output: partition $\mathcal{P} = \{X_1, X_2, \cdots X_k\}$ of V such that
  - For all $S \neq \emptyset$ and $S \subset X_i$: $|E(S, X_i \setminus S)| \geq \alpha \min(|S|, |X_i \setminus S|)$ i.e. $G[X_i]$ is an $\alpha-$expansion expander.
  - $\sum_{i=1}^{k} |E(X_i, V \setminus X_i)| = O(\alpha n \gamma)$
- Time: $O(m\gamma)$ where $\gamma = n^{o(1)}$

Let us consider any min-cut C of the graph G we will show that it either contains a whole expander of expander decomposition (or) has minimal overlap with an expander. If min-cut contains the whole expander then contracting any subset of that expander will not affect the min-cut C. Otherwise we should have minimal overlap.

This is intuitively true from the definition of expander because whenever the overlap with expander is more because of the expansion property of expander number of edges going out of the overlapped part is more but limited by size of the minimum cut which will limit the number of vertices in the overlap. Formally let $S = C \cap X$ be the intersection of the min-cut C and expander X and assume without loss of generality that S is smaller side in X if not we can consider $\overline{C}$ in G which is also a min-cut and then take the intersection with X.

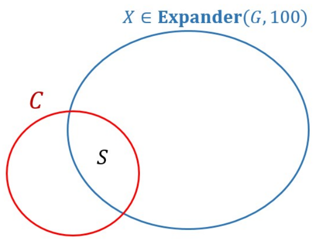

Figure 3: Mincut has small overlap with expander

Lemma 3.1. $|S| \leq \lambda/100$.

Proof.

$$\begin{aligned}\lambda &\geq E(S, X \setminus S) && (E(C, V \setminus C) \supseteq E(S, X \setminus S))\\ &\geq 100|S| && \text{(As } S \text{ is on the smaller side and } X \text{ is an 100-expander)}\end{aligned}$$

□

3.2 Trimming an Expander reduces the overlap further

Above we showed that the number of vertices in expander that a min-cut has overlap with are small and now we want to reduce that further. We see the motivation to come up with a process like $Trim$ that will help us progress towards our goal.

Observe that any vertex in min-cut C will have at most $1/2$ of it's incident edges going out because if its not the case then we can remove it from the cut C to get a smaller cut. This is the basic clue for defining a process like $Trim$. Since any vertex has at most half of it's edges going out so any vertex $v \in S$ would have at most $1/2$ of its edges coming inside of expander X (as some edges can go out of $C \cup X$). So, intuitively we can delete any vertex in X that has less than $1/2$ of its degree inside X.

We now formally define the process of $Trim$.

Algorithm 1: $Trim(X)$

```
**Initialize:** $X' = X$
**while** $\exists v \in X$ *such that* $|E(v, X)| < 2\delta/5$ **do**
    $X' = X' \setminus \{v\}$
**return** $X'$
```

After the process from the guarantee of the algorithm that every $v \in X'$ has $|E(v, X')| \geq 2\delta/5$.

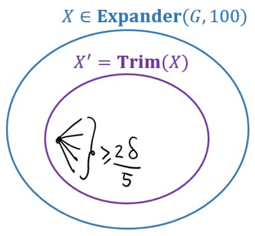

Figure 4: Guarantee after $Trim$ routine

Note that at first look it might seem that the vertices that are deleted depend on the order of deletions. However, it is not true and the set of vertices deleted does not depend on the order. This is because deletion of one of the vertex will only decrease the number of edges incident in X for the other vertex so it will be deleted sooner or later and hence any vertex that would have been deleted will be deleted and cannot escape from deletion because of a choice between vertices. It can be thought of as a dependency graph where deletion of a vertex reduces the degree of other vertex.

It can also be that the whole vertex set X is trimmed making X' empty thus giving no reduction in number of vertices using contraction. However we will later show that the number of vertices in the final graph G' would still be bounded by n/δ.

Now we show a stronger bound on the number of vertices that any min-cut C has overlap with any of the trimmed sets. Let $X' = Trim(X)$ and $S' = (X' \cap C) \subseteq S$.

Lemma 3.2. *$|S'| \leq 2$.*

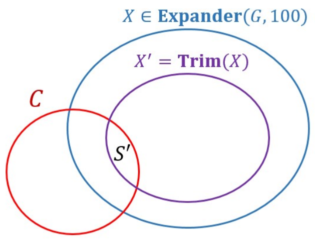

Figure 5: Overlap of Mincut after $Trim$

Why should S' be so small?

If S' is not small, because each and every vertex in S' emits at least $2\delta/5$ edges out of them and only small fraction of them can go inside S' (as $|S'|$ is bounded trivially by size of S). So, many edges go out of S'. But many edges cannot go out of S' as it is contained in min-cut C whose size can be only λ. This is formalized in the proof below.

Proof. Assume $|S'| \geq 3$ for contradiction. We also have that $|S'| \leq |S| \leq \frac{\lambda}{100} \leq \frac{\delta}{100}$. Since edges going out must be at most λ and edges emitting out from vertices in S' are at least $|S'| \cdot \frac{2\delta}{5}$ and edges that can go inside S' can be at most $\binom{|S'|}{2} \leq |S'|^2$. Here we need the requirement that G is simple graph, if not number of edges cannot be bounded as shown.

So we have contradiction as follows:

$$\begin{aligned} \lambda &\geq |E(S', X' \setminus S')| && (S' \text{ contained in min-cut } C) \\ &\geq |S'| \cdot \frac{2\delta}{5} - 2|S'|^2 \\ &\geq \delta && (\text{when } |S'| \in [3, \tfrac{\delta}{100}] \text{ for large enough } \delta \gtrsim 265) \end{aligned}$$

□

Note that the constant 265 is just the result of solving the inequality with appropriate bounds. What happens if δ is not large enough. Then we are done right!!! Why?

Because when δ is small λ is small too and we can use [Gab95] directly to find edge connectivity in $\tilde{O}(\lambda \cdot m) = \tilde{O}(m)$.

3.3 Shaving an Expander ensures no overlap

Trimming is done now lets $Shave$. Observe that in trimming as we have discussed that removing any vertex that has smaller degree inside expander X will remove the vertices that might be inside the min-cut C. However we have used a weaker requirement of deleting vertices v, which have fewer than $\frac{2\delta}{5}(< \frac{deg(v)}{2})$ degree inside expander X. This relaxation will help us to reduce the number of vertices that need to be deleted as we do not want to end up pruning the whole expander. And as we saw above that the $Trim$ is good enough, but does not completely eliminate vertices inside min-cut C.

This necessitates the need for $Shave$ process which as we will show completely removes the vertices inside C. Let X' be any set we have

Algorithm 2: $Shave(X')$

```
    $X'' = \{v \in X' : |E(v, X')| > \frac{deg(v)}{2} + 1\}$
    return $X''$
```

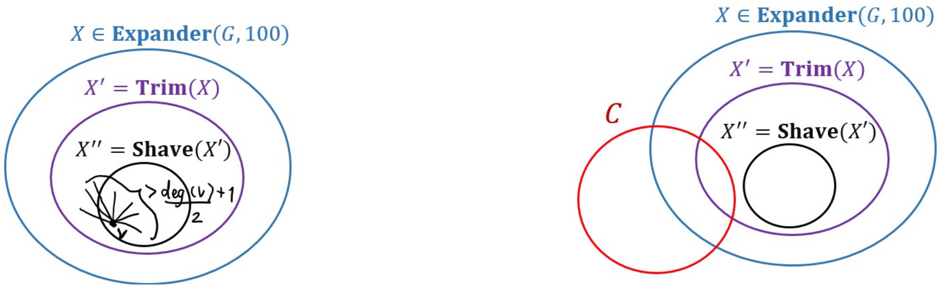

Figure 6: Mincut does not overlap after $Shave$

Let $X'' = Shave(X')$ and $S'' = (X'' \cap C) \subseteq S'$.

Lemma 3.3. $S'' = \emptyset$

Proof. Assume S'' has a vertex v and since C is a non-trivial min-cut there is at least another vertex in C. Since $|S'| \leq 2$ there can be at most another 1 vertex in S'' other than v. Since $v \in X''$ we have

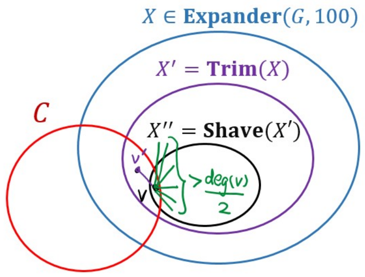

Figure 7: Non empty intersection implies a smaller min-cut

$|E(v, X')| > \frac{deg(v)}{2} + 1$ and so number of edges going out of C is at least $> (\frac{deg(v)}{2} + 1) - 1 = \frac{deg(v)}{2}$ and so we can remove v out of C still leaving behind a valid cut (because C is non-trivial min-cut) that has size smaller than min-cut which is a contradiction. □

Note it is very crucial that we need C to be a non-trivial min-cut otherwise removing v will make it empty and would not lead to contradiction.

3.4 Final steps

Hence we have shown that applying $Trim, Shave$ routines to each of expander sets in expander decomposition will ensure that the remaining sets $X'' \in P''$ will not have intersection with any non-trivial min-cut C. Since they are either completely inside (or) outside a min-cut we can safely contract them to give a smaller graph G' which preserves all non-trivial min-cuts as required.

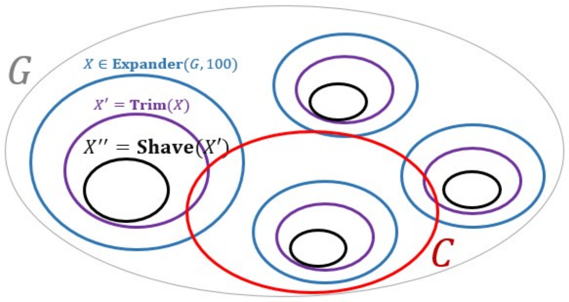

Figure 8: Shaved sets can be safely contracted

4 G' is small

As we have seen in previous section we have computed the graph G' however we still have to prove that the number of edges and vertices of G' are few so that we can use [Gab95] on this graph and still achieve the required time bound.

To start with, note that the number of inter expander edges in expander decomposition is $O(\alpha n\gamma) = O(n\gamma)$. If we have contracted all vertex sets $\mathcal{P}$ in G i.e. $G/\mathcal{P}$ then we would have got $O(n\gamma)$ edges in G'. However, we have added more edges and vertices to G' by $Trim, Shave$ routines as shown in figure below.

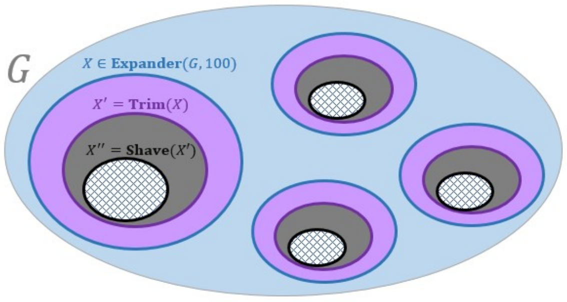

Figure 9: Edges in G' are $O(n\gamma)$

We bound the number of edges (and vertices) in the regions produced by these routines using careful charging argument and show that up to constant factors number of edges are still $O(n\gamma)$.

Then we have G' with $O(n\gamma) \le O(\frac{m\gamma}{\delta})$ edges as δ is min degree.

4.1 Why we don't trim too much?

As we proceed with deleting vertices from an expander $X \in \mathcal{P}$ which means we are reducing the number of vertices that are going to be contracted thus increasing both number of edges,vertices that are going to be finally in G'.

To bound the number of edges that are added we define a state tuple that represents the information about number of edges.

Let (t, e) represent the state of the process $X' = Trim(X)$ where X' is a dynamic set that changes with each deletion where as $X \in \mathcal{P}$ is the constant set that is part of expander decomposition. We define t as the number of boundary edges that are going out of X' and e is the number of edges that are present outside all dyanmic sets $X' \in \mathcal{P}'$ that is number of edges in $e = |E(G/\mathcal{P}')|$ as X' is dynamic so is $\mathcal{P}', e$.

Initially $X' = X$ so total count of boundary edges coming out of $X' = X \in \mathcal{P} = \mathcal{P}'$ is just the count of inter expander edges in expander decomposition. Observe that it is also the value of e which is the number of edges in $G/\mathcal{P}'$.

Observe that every vertex $v \in X'$ that is being removed has $\le \frac{2}{5}\delta \le \frac{2}{5}deg(v)$ edges inside X' whenever we delete it from X'. Also there at least $\frac{3}{5}deg(v)$ edges going out of X' from this vertex. Here comes the charging argument from the two key observations stated below.

1. Once the vertex is removed the number of boundary edges going out of X' is reduced by at least $\frac{3}{5}deg(v) - \frac{2}{5}deg(v)$ as there are at most $\frac{2}{5}deg(v)$ new edges are becoming boundary edges as now v is outside X'. Hence the number of boundary edges that are decreased are at least $\frac{deg(v)}{5}$ due to removal of v.

2. Also whenever a vertex is removed from X' there are at most $\frac{2}{5}deg(v)$ which are being added to G'. So e increases by $\frac{2}{5}deg(v)$.

From the above two observations every time a vertex is removed t decreases by at least $\frac{deg(v)}{5}$ and e increases by at most $2\frac{deg(v)}{5}$. Hence the increase in e can be charged to decrease in t and since t cannot decrease beyond zero becoming negative hence e can increase at most by 2 times the initial value of t which is $O(n\gamma)$. So total number of edges increases due to trimming

$$|E(G/\mathcal{P}') \setminus E(G/\mathcal{P})| = O(n\gamma).$$

4.2 Why we don't shave too much?

Here, we bound the size of $E(G/\mathcal{P}'') \setminus E(G/\mathcal{P}')$.

For each $X' \in \mathcal{P}'$, we add more edges to G' when we remove vertices from X' just like in trimming. Number of edges that are added to G' are at most $\sum_{v \in X' \setminus Shave(X')} |E(v, X')|$.

For each $v \in X' \setminus Shave(X')$, we have $|E(v, X')| \le \deg(v)/2 + 1$ so $|E(v, V \setminus X')| \ge \deg(v)/2 - 1$. Assuming $\delta \ge 4$, we have

$$|E(v, X')| \le 4|E(v, V \setminus X')|$$

and so

$$\sum_{v \in X' \setminus Shave(X')} |E(v, X')| \le 4|E(X', V \setminus X')|$$

$$
\begin{aligned}
|E(G/\mathcal{P}'') \setminus E(G/\mathcal{P}')| &\le \sum_{X' \in \mathcal{P}'} 4|E(X', V \setminus X')| \\
&\le \sum_{X \in \mathcal{P}} 4|E(X, V \setminus X)| = O(n\gamma)
\end{aligned}
$$

where the last inequality is because $Trim$ only decreases $|E(X, V \setminus X)|$ and so $|E(X', V \setminus X')| \le |E(X, V \setminus X)|$ for each $X' = Trim(X)$.

4.3 Size of G'

So total number of edges in G' after $Trim$ and $Shave$ are

Lemma 4.1. *$|E(G')| = O(m\gamma/\delta)$.*

Proof.

$$
\begin{aligned}
E(G') &= E(G/\mathcal{P}) \\
&\quad + E(G/\mathcal{P}') \setminus E(G/\mathcal{P}) \\
&\quad + E(G/\mathcal{P}'') \setminus E(G/\mathcal{P}') \\
|E(G')| &= O(n\gamma) && \text{(Each set has } O(n\gamma) \text{ edges)} \\
&\le O(m\gamma/\delta) && (m \ge n\delta)
\end{aligned}
$$

□

Lemma 4.2. *$|V(G')| = O(n\gamma/\delta)$.*

Proof. It is enough to show that each and every vertex in G' has degree at least δ. Vertices which are part of the original graph have degree at least δ. Vertices that are formed as a result of contraction can be categorized into two cases based on number of vertices that are contracted.

1. Number of vertices contracted $\le \frac{\delta}{2}$. Since each vertex inside contracted set has at least δ edges incident on it and at most $\frac{\delta}{2}$ remain inside the contracted set. So at least $\frac{\delta}{2}$ edges go out of the set. Since there are at least two vertices we have at least δ edges going out of the contracted set.

2. Number of vertices contracted $> \frac{\delta}{2}$ and number of such sets can be at most $O(n/\delta)$.

So total number of vertices in G' are at most $\frac{O(n\gamma)}{\delta} + O(\frac{n}{\delta}) = O(\frac{n\gamma}{\delta})$. □

References

[CGL+19] Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. *CoRR*, abs/1910.08025, 2019.

[Gab95] H.N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences*, 50(2):259–273, 1995.