Towards the Erdős–Gallai cycle decomposition conjecture

Talk by Matija Bucic
Notes by Sanjana Das
November 4, 2022

This is joint work with Richard Montgomery.

§1 Introduction

§1.1 Graph decomposition

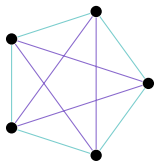
Definition 1.1. A decomposition of a graph is a partition of its edge set.

Question 1.2. Given a graph, can we decompose it into a 'few' graphs with certain 'nice' properties?

This question, and its generalization to hypergraphs, has many applications in various areas. In some sense, when we want to understand a large structure, a natural way to do that is to decompose it into smaller substructures with nice properties (e.g., triangles or cycles) — because then we can understand those smaller structures better, and we can hope to lift that understanding back to the larger structure.

Today we'll be focusing on decomposing graphs into *cycles*.

Theorem 1.3 (Walecki 1883)
The complete graph K_{2n+1} can be decomposed into n cycles.



Note that this is the optimal number of cycles (i.e., we can't use fewer than n), since K_{2n+1} has $n(2n+1)$ edges, and each cycle uses at most $2n+1$ of them.

Taking a step back, there's an even more natural question.

Question 1.4. What kinds of graphs can be decomposed into cycles?

One thing that guarantees a graph *can't* be decomposed into cycles is having a vertex of odd degree — every cycle contributes degree either 0 or 2 to every vertex. (This is why Theorem 1.3 is for K_{2n+1} and not K_{2n}.) But it turns out that this is the only obstruction.

Theorem 1.5 (Veblen 1912)
Any graph with all degrees even can be decomposed into cycles.

This follows from an even older observation that any graph with all degrees even has an Eulerian circuit — an Eulerian circuit might self-intersect, but we can break it up at these self-intersections to get a decomposition into cycles.

§1.2 The Erdős–Gallai conjecture

In some sense, it's natural to ask whether we can combine Theorems 1.3 and 1.5 — Theorem 1.5 tells us when it's *possible* to decompose a graph into cycles, and Theorem 1.3 is about the *optimal number* of cycles in such a decomposition (specifically for odd cliques). The Erdős–Gallai conjecture is one attempt to do so.

Conjecture 1.6 (Erdős–Gallai) — Every n-vertex graph can be decomposed into $O(n)$ cycles and edges.

Here we allow our decomposition to also contain some lone edges, to get rid of the even-degree condition. But this conjecture is equivalent to the statement that every n-vertex Eulerian graph (i.e., graph with all degrees even) can be decomposed into at most $O(n)$ cycles, which was conjectured by Hajós (in fact, Hajós even conjectured that any such graph can be decomposed into $n/2$ cycles).

One difference between this problem and many other graph decomposition problems is that in many problems, it's not hard to prove some linear bound, and the main difficulty is nailing down the correct constant. But here we don't even know how to get a linear bound, despite 60 years and many tries.

If the conjecture is true, then the bound of $O(n)$ is tight — for example, any graph with all degrees odd needs at least $n/2$ edges in the decomposition (in general, every odd-degree vertex needs an edge). There are constructions with better constants — Gallai (1966) had a construction requiring $(4/3 - o(1))n$ cycles and edges, and Erdős (1983) had one requiring $(3/2 - o(1))n$.

§1.3 Some related problems

Lovász (1968) proved that the conjecture is true if we want to decompose into *paths* rather than cycles. The proof is a very nice three-page induction argument, and it's the starting point for lots of arguments in this area. There's also a conjecture of Gallai asserting the correct constant for this theorem; and there's a long list of graphs for which this conjecture has been proved (for example, there's a 100-page paper proving it for planar graphs).

But paths are easier than cycles. One reason for this is that paths (and trees) have natural traversals, but cycles don't — we need to remember where our start was. So this makes cycles a natural difficulty threshold, since if we want to be able to decompose a graph into cycles, then as we're embedding a cycle into the graph, we need to remember the history of this embedding.

A second related problem is where we want to *cover* rather than *decompose* the graph — so now our cycles are allowed to overlap. This problem was solved by Pyber (1985); then there were two conjectures strengthening it, solved by Fan in the early 2000s. This problem is highly nontrivial, but it's understood much better than the decomposition version.

§1.4 Some special cases

There are two large classes of graphs for which we know the Erdős–Gallai conjecture is true. The first such class is graphs with linear minimum degree — here the result was proved by Conlon, Fox, and Sudakov (2013), and the correct leading constant was nailed down in 2021 (it's 3/2).

The other class of graphs for which we know the conjecture is true is *quasirandom graphs*. (We're not going to go into what quasirandom graphs are.) Again the conjecture was proved by Conlon, Fox, and Sudakov, and the correct asymptotics have also been nailed down; Glock, Kühn, and Osthus nailed down a (more) exact result in 2016.

So if we have a little bit of structure (specifically, large minimum degree), then we can use this structure to prove the conjecture. And if we have a lot of randomness, then we can use the tools we have for understanding random objects, and again the conjecture is slightly easier. The difficult case is when we have neither this bit of structure nor randomness.

§1.5 The general case

We'll now discuss bounds for the general case (on the number of cycles and edges needed), starting with the following simple bound.

Theorem 1.7 (Folklore)
Any n-vertex graph can be decomposed into $O(n \log n)$ cycles and edges.

Proof. We first take the longest cycle and add it to the decomposition (and delete it from the graph). Then we find the longest cycle in the remainder and add it to the decomposition as well. We keep on doing this until the graph is a forest, at which point there's at most $n-1$ edges. It's not hard to show that if G has average degree d, then this process runs for $O(n \log d)$ steps (by considering how the average degree decreases at each step). □

It took 50 years for this bound to be improved, from $\log n$ to $\log \log n$.

Theorem 1.8 (Conlon–Fox–Sudakov)
Any n-vertex graph can be decomposed into $O(n \log \log n)$ cycles and edges.

Our main result improves this bound further, replacing $\log \log n$ with $\log^* n$ (where $\log^*$ is the iterated olgarithm function — the number of times we need to take a logarithm to make n less than 1).

Theorem 1.9 (Bucic–Montgomery 2022)
Any n-vertex graph can be decomposed into $O(n \log^* n)$ cycles and edges.

Remark 1.10. There were two results in between — between Theorems 1.7 and 1.8 we had a bound of $(n \log n)/\log \log n$, and between Theorems 1.8 and 1.9 we had a bound of $(n \log \log n)/\log^* n$. Interestingly, Theorems 1.8 and 1.9 both 'flipped' these results (respectively), moving the denominator into the numerator.

We'll now discuss the ideas behind the proof of Theorem 1.9.

§2 A proof overview and path decomposition

We're going to make use of the theorem of Lovász on decomposing a graph into *paths*, which is at the heart of a large number of results on sparse object decomposition.

Theorem 2.1 (Lovász 1968)

Any n-vertex graph can be decomposed into at most $n/2$ paths and cycles.

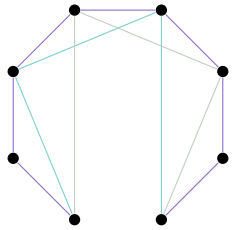
We're actually going to use a slight modification of this result. But first, to motivate what this modification is (and certain future lemmas), we'll vaguely discuss how the argument will work.

We're starting with some arbitrary graph, and we want to decompose it into cycles. To do so, we're first going to decompose it into very structured objects (specifically expanders); and then we're going to reduce the problem of decomposing the *entire* graph to just decomposing these expanders. So now it remains to decompose just these well-structured objcts.

We'll have a lemma that such an object is well-structured because of a sparse subgraph — i.e., there's a sparse subgraph that's responsible for all the expansion. Then we want to use this sparse substructure as an *absorber* — we set it aside and save it to fix some issues later, and first try to decompose everything except this sparse substructure. We'll use the theorem of Lovász to decompose this leftover graph into *paths*, and we'll then use the special sparse substructure to join up these paths into actual cycles.

§2.1 A modified path decomposition result

There are quite a few issues with this outline, but the first one that comes to mind is that in the result of Lovász, we don't have any control on where the endpoints of these paths go — so if we decompose a somewhat dense graph, it's possible that all the endpoints of the paths are in the same place.

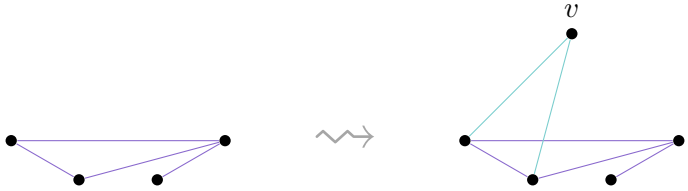


This would be horribly bad for our absorption argument, because the absorber needs to be sparse — we need to deal with the majority of the graph using Lovász, and it's a problem if we can't control the endpoints of the paths it gives. So we're going to prove a modification that lets us actually ensure the endpoints are well-spread around the graph, so that the sparse substructure that we set aside can deal with them.

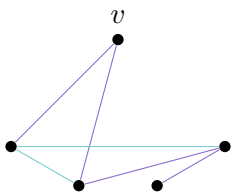
Corollary 2.2

Every n-vertex graph can be decomposed into at most n paths such that no vertex is an endpoint of more than two paths.

Proof. We first add an auxiliary vertex v and join it to all even-degree vertices in the original graph; now we have a graph where the n original vertices all have odd degree.



Now we use Lovász's theorem to decompose this graph into $(n+1)/2$ paths and cycles.



Since we had n odd-degree vertices and every odd-degree vertex has to be the endpoint of some path, there must be at least n endpoints. But we only have $(n+1)/2$ paths and cycles in this decomposition, and if we used even one cycle then we'd have at most $n-1$ endpoints; so this must actually be a decomposition into *paths* (with no cycles). Similarly, each of the n original vertices is the endpoint of exactly one path — otherwise it would have to be the endpoint of at least three paths (for parity reasons), and we'd be left with at most $n-2$ endpoints to deal with the remaining $n-1$ points.

So this magically forces the decomposition to work nicely and be well-spread around the graph — specifically, each of the original vertices must be the endpoint of exactly one path. Now if we remove our extra vertex v, then each vertex becomes the endpoint of at most one additional path — this is because removing v splits each path into at most two pieces, and the new endpoints are the vertices which were originally adjacent to v along this path (and every vertex can only have been adjacent to v along at most one edge, so it becomes the endpoint of at most one new path). □

Remark 2.3. This argument relies on the fact that Lovász's theorem gives exactly $n/2$ objects — this is the optimal possible number, and it gives us a lot of power.

§3 Robust sublinear expanders

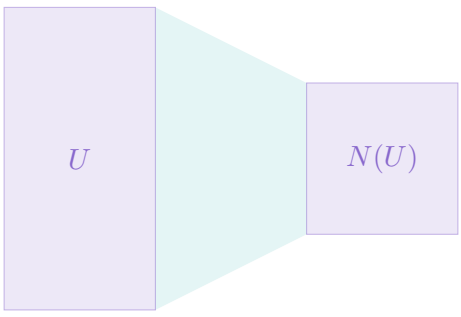
As mentioned earlier, our goal is to decompose our graph into expander-like structures.

§3.1 Sublinear expanders

First, here's the usual notion of an expander.

Definition 3.1. For a vertex subset U, we use $N(U)$ to denote its external neighborhood, the set of vertices *outside* U which are adjacent to U.

Definition 3.2. We say a n-vertex graph is a λ-expander if every vertex subset U of size at most $n/2\lambda$ has an external neighborhood of size $|N(U)| > \lambda |U|$.



Definition 3.3. We say a n-vertex graph is a sublinear expander if every vertex subset U of size at most $n/2$ has an external neighborhood of size $|N(U)| > |U|/(\log n)^2$.

As mentioned in the overview, the first step of our proof is to reduce the graph to expanders. If we're aiming for expanders with *constant* λ, there's no chance of being able to guarantee that we can find such an expander in our graph. So that's why we work with this weaker notion of *sublinear* expanders, where the factor of expansion decreases with n. It's a fact that we *can* always find a subgraph satisfying this weaker notion of expansion — and importantly, we can do this while preserving the average degree.

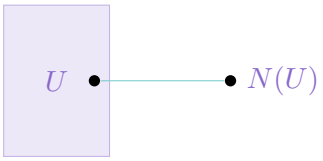
Theorem 3.4

Given any graph, we can find a subgraph which is a sublinear expander of roughly the same average degree as the original graph.

This is really useful — it means that for many problems, we can essentially assume our graph is a sublinear expander for free.

§3.2 Robustness

Unfortunately, sublinear expansion doesn't give as much power as we'd like — there are nice properties of expanders that break down for sublinear expanders. Specifically, expanders are very well-connected — they're not only connected, but also resilient to the removal of edges. But for a *sublinear* expander, if we consider a set U of size $(\log n)^2$, the condition of sublinear expansion only requires it to have *one* external neighbor, and it's possible that this one neighbor has just one edge going back to U. And then if we delete this one edge, we've disconnected our sublinear expander.



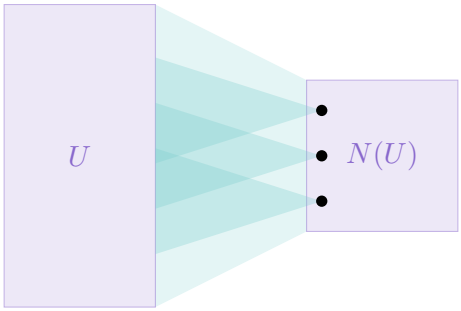
So sublinear expanders don't have any type of robustness — they're connected, but it's very easy to disconnect them (they're not resilient to removing edges). And this is problematic.

In general, we can't hope to get better expansion than $1/(\log n)^2$ if we want a statement like Theorem 3.4 to hold. But it turns out that we *can* add more conditions in parallel that ensures we don't have this issue — that the expander can't be easily disconnected by removing edges.

Definition 3.5. We say a n-vertex graph G is a d-robust (sublinear) expander if for all $|U| \leq n/2$, at least one of the following two statements holds:

- We have $|N(U)| > d|U|$.
- $N(U)$ contains at least $|U|/(\log n)^2$ vertices which have at least $d(\log n)^2$ neighbors in U.

In the first case, we get much better expansion — we get expansion by a constant factor d, as compared to $1/(\log n)^2$. This is the 'golden case,' and it's reasonably easy to deal with. In the second case, we don't get better expansion — we're still only guaranteed expansion by a factor of $1/(\log n)^2$ — but we're guaranteed lots of edges going back, which makes this more robust (e.g., harder to disconnect).



Note that in both cases, the number of edges we're guaranteed going back from $N(U)$ to U is at least $d\,|U|$ — in the first case, we're even guaranteed at least $d\,|U|$ *vertices* in $N(U)$, and each such vertex has at least one edge. Meanwhile, in the second case, we're again guaranteed at least

$$\frac{|U|}{(\log n)^2} \cdot d(\log n)^2 = d\,|U|$$

edges going back. So in some sense, the notion of a robust expander combines the ideas of vertex and edge expansion.

§3.3 Some lemmas on robust sublinear expanders

We'll now discuss some facts about robust sublinear expanders. Note that sublinear expanders are a *weaker* notion than regular expanders, but the robustness gives us some extra power; it's a crucial ingredient, and most of these lemmas would fail without it.

The first lemma states that we can almost partition an arbitrary graph into robust sublinear expanders.

Lemma 3.6

For any d, given any n-vertex graph, we can partition all but $n \cdot \mathsf{polylog}(n)$ of its edges into d-robust sublinear expanders $H_1, \ldots, H_t$. Furthermore, we can ensure that

$$|V(H_1)| + \cdots + |V(H_t)| \leq 2n.$$

The more edges we're willing to sacrifice, the more robustness we can guarantee (i.e., we can make d as large as we want by sacrificing a sufficiently large polylog factor). In fact, on the other end, we can get a *perfect* decomposition lemma (one with no missing edges at all) if we take $d = 0$ — we can decompose any graph into sublinear expanders. This in particular means that if we could prove the Erdős–Gallai conjecture for sublinear expanders, then we could prove it in general.

> **Remark 3.7.** Note that d doesn't affect the vertex expansion of our sublinear expanders — it only affects their robustness (in particular, $d = 0$ corresponds to a sublinear expander with no robustness).

The second feature of Lemma 3.6 is that when we partition into expanders, we can do so with very little vertex overlap — a typical vertex is only in at most two of our expanders. (This will be crucial when we reduce from decomposing the entire graph to decomposing our expanders.)

The proof of Lemma 3.6 is fairly simple — if our current graph has a subset that fails the robust expansion condition, then this subset defines a somewhat sparse-ish cut, and we can reduce the problem to dealing with the two sides of this cut. (This is a standard trick for how you find expanders in general, and if you do a careful calculation, you can ensure the desired dependencies.)

The next lemma states that if we randomly edge-sample a robust sublinear expander, then it remains one.

Lemma 3.8

If we take a robust sublinear expander and keep every edge with probability $1/2$, it remains a robust sublinear expander with reasonably high probability.

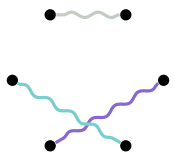
This can be proven by a union bound, and it's not hard.

One consequence of Lemma 3.8 is that it means we can split one robust sublinear expander into several (by randomly assigning each edge to one of the subgraphs). This is nice because it allows us to parallelize some parts of our argument. Specifically, we'll eventually have a bunch of points that we want to join with short paths (once we've decomposed the rest of the graph into paths using Lovász, and we're trying to complete those paths into cycles). This will essentially allow us to split those points into batches, and use a separate subgraph to deal with each batch. (Lemma 3.8 is actually used in three different proofs; this is one of them.)

In more detail, we have the following property, which states that these robust sublinear expanders have very strong connectivity — if we take a robust sublinear expander with sufficiently large d and an adversary gives us a bunch of disjoint pairs of points to connect with paths, then we really can connect those pairs with short edge-disjoint paths.

Theorem 3.9

Suppose we have a d-robust sublinear expander (with d sufficiently large). Then given any collection of linearly many disjoint pairs of vertices, we can find short edge-disjoint paths joining each pair.



Standard expansion arguments would allow us to connect $\mathsf{polylog}(n)$ pairs of points in this way, but Theorem 3.9 says that we can actually connect *linearly* many.

Then Lemma 3.8 makes our life easier when we use Theorem 3.9 to join up our paths into cycles — Corollary 2.2 to Lovász's theorem says that we can perform the path decomposition such that each vertex is the endpoint of at most two paths, which means it's in at most two of our pairs that we want to connect. Then we can split these pairs into batches such that each batch has no repeated vertices, use Lemma 3.8 to split up our robust sublinear expander into several ones (with one for each batch), and apply Theorem 3.9 separately to each. So Lemma 3.8 makes our life easier when we use strong connectivity (Theorem 3.9).

This is a strong statement, but it's not strong enough — in fact, Theorem 3.9 remains true if we also insist that our paths go through a random subset of vertices.

Lemma 3.10

Suppose we have a d-robust sublinear expander (with d sufficiently large), and we sample a random subset of vertices with probability $1/3$. Then with high probability, given any collection of linearly many disjoint pairs of vertices, we can find short edge-disjoint paths joining each pair.

(The proof is a nice random process argument.)

The final lemma we'll need is on the existence of an expanding *skeleton* — it essentially states that there's a sparse witness of the expansion.

Lemma 3.11

Given an n-vertex d-robust sublinear expander, we can find a subgraph with $n \cdot \mathsf{polylog}(n)$ edges and with the same expansion properties.

So the original expander itself might be fairly dense, but we can find a sparse-ish subgraph which inherits the same expansion property — this means the expansion properties of our original graph are really because of some sparse subgraph, rather than being a property of *everything* in the graph.

The way we prove Lemma 3.11 is by using the template method — we use a random graph as a template, and there's a neat trick of how we can use strong connectivity (Theorem 3.9) and parallelization (as in Lemma 3.8) to find such a skeleton in an *arbitrary* robust sublinear expander.

§4 Proof of Theorem 1.9

We'll now discuss the proof of Theorem 1.9. We'll actually prove the following *asymmetric* version.

Theorem 4.1

Any n-vertex graph H can be decomposed into $6n$ cycles and $n \cdot \mathsf{polylog}(n)$ edges.

Here we're sacrificing $n \cdot \mathsf{polylog}(n)$ edges. But we're allowed to pay this because we can iterate — we first use Theorem 4.1 to remove a bunch of cycles from our graph and be left with $n \cdot \mathsf{polylog}(n)$ edges. Then we iterate on those edges to decompose them into a bunch of cycles and $n \cdot \mathsf{polylog}(\log n)$ edges, and so on. (This iteration isn't immediate from Theorem 4.1 — for this to work, we actually need a version where the $\mathsf{polylog}$ factor depends on the average degree, not n — but we'll mention that statement later.)

Proof. First, we'll reduce to the case of decomposing a robust sublinear expander — suppose we know that Theorem 4.1 is true when H is a robust sublinear expander. Then given an arbitrary graph H, we start by decomposing all but $n \cdot \mathsf{polylog}(n)$ of its edges into robust sublinear expanders $H_1, \dots, H_t$ such that

$$|V(H_1)| + \cdots + |V(H_t)| \leq 2n.$$

We then apply the theorem to each H_i (we'll actually prove the theorem with 6 replaced by 3 in the case of a robust sublinear expander — meaning that each H_i can be partitioned into at most $3\,|V(H_i)|$ cycles and $|V(H_i)| \cdot \mathsf{polylog}(|V(H_i)|) \leq |V(H_i)| \cdot \mathsf{polylog}(n)$ edges). This gives us a decomposition of the entire graph into at most

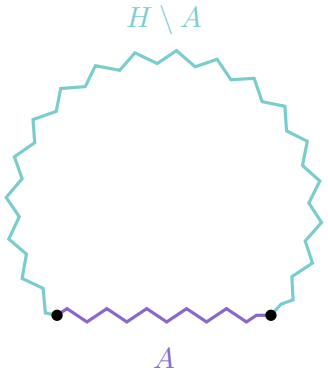
$$3\,|V(H_1)| + \cdots + 3\,|V(H_t)| \leq 6n$$

cycles, and similarly at most $(|V(H_1)| + \cdots + |V(H_t)|) \cdot \mathsf{polylog}(n) = n \cdot \mathsf{polylog}(n)$ edges.

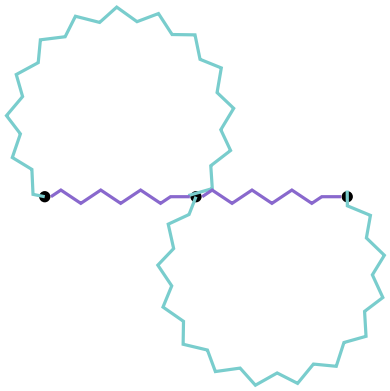
So now it suffices to consider the case where H is a robust sublinear expander (with large d).

We first set aside a sparse expanding skeleton A of H with $n \cdot \mathsf{polylog}(n)$ edges (using Lemma 3.11). Note that our decomposition is allowed to sacrifice $n \cdot \mathsf{polylog}(n)$ edges, so it doesn't really matter how much of A the decomposition handles — it's enough to decompose the rest of the graph, and the remainder of A that we haven't decomposed can go into these extra $n \cdot \mathsf{polylog}(n)$ edges.

To get this decomposition of the rest of the graph, we can first use Corollary 2.2 to decompose $H \setminus A$ into linearly many paths. And then we can use strong connectivity (from Theorem 3.9) to close these paths into cycles (by connecting the endpoints of each path with a short path inside A).

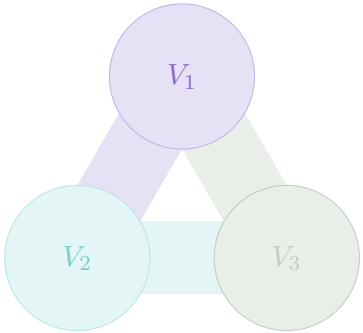


If there were no issues with this, the paper would be much shorter. But there actually is a caveat — the issue is that the paths we get from Corollary 2.2 could potentially share *vertices* with the paths we use in our skeleton A to connect their endpoints.

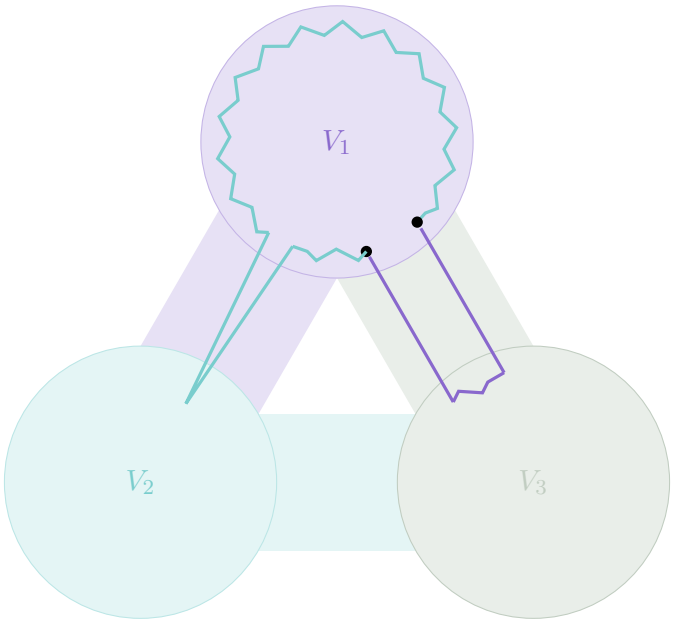


If this happened, then when we connected the endpoints in such a situation, we'd get a closed walk but not necessarily a cycle — we'd have to split it into *multiple* cycles, and this would be too costly (we *do* know that the paths in A are reasonably short, but they're not so short that we can afford to lose a factor that scales with their length).

To fix this, we need to ensure that our long paths live in a separate part of the graph (in terms of vertex subsets) than the corresponding short paths. And this is what the random sampling mentioned in some of the earlier lemmas is for. We first take our expanding skeleton A and split its edges into three expanding skeletons A_1, A_2, and A_3 (using Lemma 3.8) — we'll use these skeletons to deal with separate parts of the graph. We then take our *vertices* and split them into three sets V_1, V_2, and V_3 (uniformly at random). And we decompose $H \setminus A$ into three pieces (which we think of as the colors *red*, *blue*, and *green*) — we color an edge red if it is either within V_1 or between V_1 and V_2, blue if it is either within V_2 or between V_2 and V_3, and green if it is either within V_3 or between V_3 and V_1.



Now we'll use Corollary 2.2 to decompose each of these three pieces into paths individually — we first use Corollary 2.2 to decompose the red edges into paths. Then since the red edges all live in $V_1 \cup V_2$ (which is disjoint from V_3), we can join up their endpoints using paths that go through V_3 to turn them into cycles — we can do this using Lemma 3.10 (as V_3 is a random set). Then our long paths (the ones from Corollary 2.2) live in $V_1 \cup V_2$ and our short paths (the ones used to connect their endpoints) go solely through V_3, so they don't intersect; and we really do get cycles.



We then do the same with the blue and green edges. Then our decomposition uses up to n cycles for each of these three pieces, so it uses at most $3n$ cycles in total, as desired.

(This decomposition uses all the edges in $H \setminus A$, but it doesn't necessarily use all the edges in A — there'll be some leftover, but that can be absorbed into the extra $n \cdot \mathsf{polylog}(n)$ leftover edges.) □

To prove Theorem 1.9, the idea is that we then iterate this argument, keeping track of the average degree (the polylog factor that we pay actually depends on this average degree, rather than n). Specifically, we can prove the following statement.

Theorem 4.2

For any constant k, any n-vertex graph with average degree d can be decomposed into $O(kn)$ cycles and $O(n \log \log \cdots \log d)$ edges (with k log's).

Then this finally gives us a decomposition with $O(n \log^* d)$ cycles and edges, as desired.