Seeded embeddings and Ramsey numbers of bipartite graphs of sublinear bandwidth

Talk by Konstantin Tikhomirov

Notes by Sanjana Das

September 26, 2024

This is joint work with Dylan Altschuler and Han Huang.

§1 Ramsey numbers of sparse graphs

This talk is about Ramsey numbers of sparse graphs, so we won't discuss diagonal Ramsey numbers (which is also an active research area).

Definition 1.1. For an undirected graph H, we define its Ramsey number, denoted $r(H)$, as the smallest N such that any 2-coloring of K_N has a monochromatic copy of H.

§1.1 Bounds in terms of maximum degree

There are several upper bounds on $r(H)$ in terms of the maximum degree of H, which we denote by Δ. First, there's the following breakthrough result from 1983, which states that if Δ is constant, then $r(H)$ is linear in the number of vertices of H (which we denote by n).

Theorem 1.2
We have $r(H) \leq C_\Delta n$ (for some constant C_Δ).

Then the next question is, what's the dependence of C_Δ on Δ? The best result is the following:

Theorem 1.3 (Conlon–Fox–Sudakov)
We can take $C_\Delta = \exp(C\Delta \log \Delta)$.

This is almost sharp — there's a matching construction of a graph H with Ramsey number

$$r(H) \geq \exp(c\Delta) \cdot n.$$

There's also results for Ramsey numbers of *bipartite* graphs. The main difference is that here, the dependence of C_Δ on Δ is exponential, rather than superexponential — it's known that

$$r(H) = O(\Delta 2^\Delta \cdot n).$$

This is also sharp up to the base of the exponent:

Theorem 1.4 (Graham–Rödl–Rucinski)
There is a bipartite graph H with maximum degree Δ and $r(H) \geq \exp(c\Delta) \cdot n$.

(We'll refer to this construction later, because it also has low bandwidth.)

Another set of results we'll consider is about d-degenerate graphs.

Definition 1.5. We say H is d-degenerate if every subgraph of H has a vertex with degree at most d. We call the smallest such d the degeneracy constant of H.

Note that every graph with maximum degree Δ is Δ-degenerate, so this is a relaxation of the maximum degree condition.

Using randomized constructions, we can construct graphs H with

$$r(H) \geq \exp(cd) \cdot n.$$

It was conjectured by Burr and Erdős that graphs with constant degeneracy numbers have Ramsey numbers linear in n, and this was proven by Lee.

Theorem 1.6 (Lee 2015)
If H is bipartite and has n vertices and degeneracy constant d, then

$$n \leq r(H) \leq 2^{Cd} n.$$

Remark 1.7. For general (not necessarily bipartite) graphs, we still get a constant depending only on d (in place of 2^{Cd}), though the dependence is worse than exponential in d.

§1.2 More precise estimates

So we have a complete description of $\log r(H)$ in terms of the number of vertices of H and its maximum degree or degeneracy constant. This is very nice, and in some sense you can say that it solves the problem for bipartite graphs. But you can still try to show more precise estimates, and that's the main content of this talk.

One graph that has received special consideration is the cube Q_k.

Conjecture 1.8 (Burr–Erdős) — We have $r(Q_k) = O(2^k)$.

This conjecture states that the Ramsey number of Q_k is linear in its number of vertices (which is 2^k). The best upper bound we know is $r(Q_k) \leq 2^{2k-ck}$ for some small constant c.

More generally, as mentioned earlier, there are random constructions of graphs with maximum degree Δ with $r(H) \geq \exp(c\Delta) \cdot n$; these graphs will have good expansion properties. On the other extreme, we can construct examples of Δ-regular graphs just by taking unions of bipartite complete graphs; if $\Delta \leq c \log n$, then these graphs will have linear-sized Ramsey numbers. So when $\Delta \leq c \log n$, in principle we could expct the Ramsey number to be anywhere between n and $\exp(c\Delta) \cdot n$.

Question 1.9. Can we understand what properties of a bipartite graph H cause $r(H)/n$ to be exponential in the maximum degree (or degeneracy constant), or on the other extreme, what properties of H cause $r(H)/n$ to be bounded above by a constant?

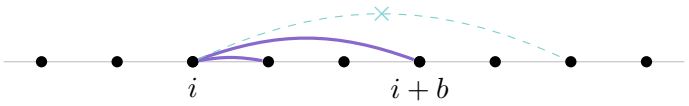
(We normalize by n because when we're considering sparse graphs — where $\Delta \leq c \log n$ — we can expect that the main contribution to $r(H)$ just comes from the number of vertices, and then we have a second-order contribution that's controlled by other properties of H.)

§1.3 Bandwidth

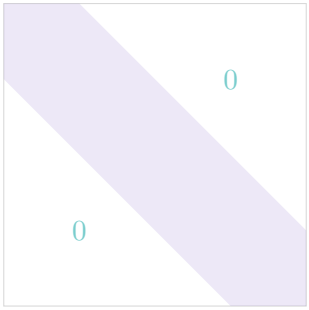
We have expander graphs that achieve close to the largest possible Ramsey number, while a disjoint union of smaller graphs has linear Ramsey number. So if we're looking for some simple conditions under which we can get upper or lower bounds, we probably want to look at something related to expansion properties.

There are several expansion properties you can consider — for example, spectral characteristics or vertex expansion. The property we'll consider is graph *bandwidth*.

Definition 1.10. The bandwidth of H is the smallest integer b such that we can order the vertices of the graph in such a way that any two vertices with $|i - j| > b$ are not adjacent.



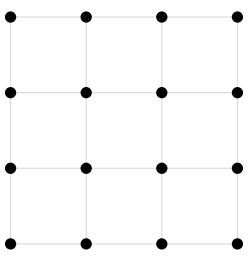
Equivalently, this means you can arrange the vertices such that in the adjacency matrix, all 1's are in a width-b region around the diagonal (and everywhere else you have 0's).



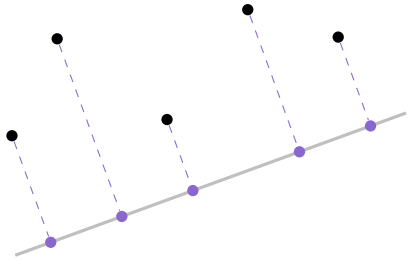
Here's a construction of graphs with small bandwidth:

Example 1.11

A n-vertex grid (of dimensions $\sqrt{n} \times \sqrt{n}$) has bandwidth $O(\sqrt{n})$ — we can order the vertices by going through the columns one at a time.



More generally, if we have a bipartite graph in the plane with low (e.g., sublogarithmic) maximum degree, then we can show that it has polynomially small bandwidth by taking a random direction, projecting our vertices onto that direction, and going from left to right (to get an ordering).



(The same works in any constant dimension.)

Question 1.12. Can we estimate the Ramsey number of a bipartite graph in terms of both its maximum degree and its bandwidth?

(We can think of the maximum degree Δ as measuring the sparsity, and the bandwidth β as measuring the connectivity.)

Definition 1.13. For each n, β, and Δ, we define $\mathcal{H}(n,\beta,\Delta)$ as the family of n-vertex bipartite graphs with maximum degree Δ and bandwidth at most βn.

Then more formally, our goal is to understand

$$\max_{H\in\mathcal{H}(n,\beta,\Delta)} \log \frac{r(H)}{n}.$$

(The reason we take a log is that we're interested in distinguishing exponential from subexponential behavior; and that corresponds to linear vs. sublinear growth of the log.)

§1.4 History and main result

From the same random construction discussed earlier, it can be shown that

$$\max_{H\in\mathcal{H}(n,\beta,\Delta)} \log \frac{r(H)}{n} = \Theta(\Delta) \quad \text{for } \beta \geq \exp(-c\Delta). \tag{1.1}$$

On the other hand, Lee (2015) showed that

$$\max_{H\in\mathcal{H}(n,\beta,\Delta)} \log \frac{r(H)}{n} = \Theta(1) \quad \text{for } \beta \leq \exp(-C\sqrt{\Delta \log n}). \tag{1.2}$$

There's also an asymptotic estimate that for every Δ, there exist n_0 and β_0 such that

$$\max_{H\in\mathcal{H}(n,\beta,\Delta)} \log \frac{r(H)}{n} = \Theta(1) \quad \text{for } n \geq n_0 \text{ and } \beta \leq \beta_0. \tag{1.3}$$

(In other words, for any Δ, if β is sufficiently small with respect to Δ, then our Ramsey numbers are linear.) This result is based on the regularity lemma, so if you convert it to a non-asymptotic estimate, the dependence between Δ and β is very bad (i.e., tower-type).

So the range of parameters $\exp(-C\sqrt{\Delta \log n}) \leq \beta \leq \exp(-c\Delta)$ hasn't been covered; the main result solves this to a large extent.

Theorem 1.14 (Altschuler–Huang–Tikhomirov 2024+)
If $\beta \le \exp(-C\Delta \log \Delta)$, then

$$\max_{H \in \mathcal{H}(n,\beta,\Delta)} \log \frac{r(H)}{n} = \Theta(1).$$

The discrepancy between this bound and the bound (1.1) from the random construction is the extra $\log \Delta$ in the exponent. Fortunately, we can use computer science notation to say that this quantity transitions from linear (in Δ) to constant at $\beta = \exp(-\widetilde{\Theta}(\Delta))$ — we don't know what's happening for such values of β, but all the other cases are settled.

§2 Seeded embeddings

We'll now discuss the proof ideas. The asymptotic statement 1.3 uses regularity; Lee's result 1.2 uses a version of dependent random choice, and so does Theorem 1.14.

The first step is fairly standard: we can reduce the problem of computing Ramsey numbers to a question about graph embeddings. Specifically, if we can embed our bipartite graph H into every bipartite graph of edge density at least $\frac{1}{2}$ with N vertices on each side, then we can get $r(H) \le 2N + 1$.

So we'll work with this embedding question instead: now we have a host graph Γ with edge density $\frac{1}{2}$ (more generally, we can imagine having any constant edge density). And our goal is to embed H into Γ.

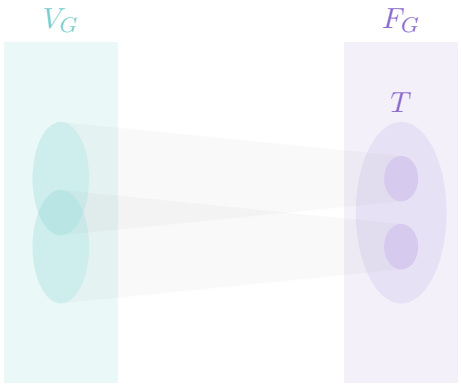
Our argument first has a preprocessing step where we find a large subgraph G in Γ which has some nice properties — we require some condition on its minimum degree, and on the cardinalities of the sets of common neighbors of small tuples. Then once we have this preprocessed graph G, we'll construct an embedding of H into G using a *seeded* version of dependent random choice.

§2.1 Dependent random choice

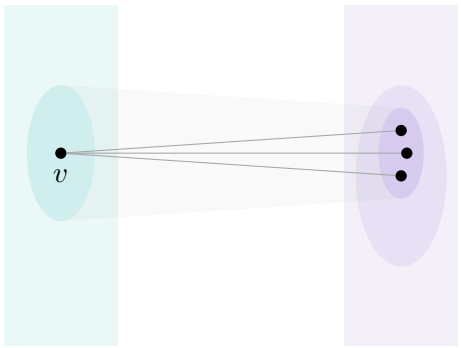
First, we'll talk about how standard dependent random choice works. Suppose we want to embed H into another bipartite graph G, where G has sufficiently many vertices and reasonably large edge density (so that it's plausible an embedding could exist).

We'll write $H = (V_H, F_H, E_H)$ and $G = (V_G, F_G, E_G)$ — so the letters V and F denote the sets of vertices on the left and right, and the letter E denotes the set of edges.

Dependent random choice works in two stages. First, we construct a subset of vertices $T \subseteq F_G$ such that for most Δ-tuples of vertices in G, their number of common neighbors in V_G is very large (e.g., at least n).



Then once we've found this set T, we map F_H (the right vertex set of H) into T uniformly at random. Then we can map the left vertices of H into G in a way that preserves adjacency — for each $v \in V_H$, we can look at where in T we've embedded its at most Δ neighbors, and embed v somewhere in the common neighborhood of those Δ vertices. (The fact that common neighborhoods of Δ-tuples in T are large allows us to do this greedily.)



Usually, the way we construct T is by first choosing a small (maybe even a constant) number of vertices in V_G uniformly at random, and then defining T as their common neighborhood. We can show this works by applying the first moment method and double-counting — by a first moment argument, we can estimate

$$\mathbb{E}[T] \approx |F_G| \cdot \alpha^K,$$

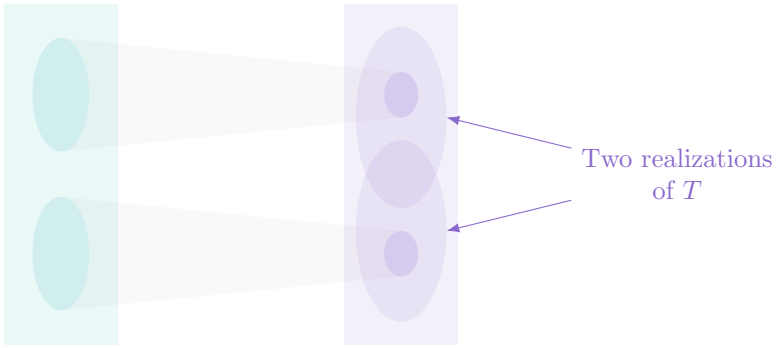
where K is the number of vertices in V_G we chose and α is the edge density. So with reasonable probability (over our choice of K points), T is not too small. We can also check that a typical Δ-tuple in T will have lots of common neighbors on the left, using a double-counting argument — if we have a Δ-tuple on the right with very few common neighbors on the left, then it's very unlikely to appear in T (because we chose T as the common neighborhood of K random vertices).

§2.2 Motivation for seeded dependent random choice

There are some simple randomized constructions that tell you dependent random choice sometimes isn't as efficient as you would like. So we'll consider a modification called *seeded* dependent random choice.

The idea of seeded dependent random choice is to not just consider one set T, but *many* sets T. In standard dependent random choice, we chose one small set of K vertices, and used that to generate one set $T \subseteq F_G$. Our modification is to instead generate *many* subsets of F_G, and define them such that different 'parts' of F_H will get mapped to different subsets. This is where we make use of the low bandwidth of H — it means that F_H can be split into a few parts which are almost not connected (there's only local connections). So we can imagine mapping each of those parts into a separate realization of T (i.e., one of these subsets of F_G), and then try to glue them together.

In principle, the advantage of this approach is that the different realizations of T are chosen independently. So if we look at Δ-tuples in different parts, their common neighborhoods are going to be sort of independent as well, which means they should have very small overlap. The problem with standard DRC is that when you construct T and look at typical Δ-tuples and their common neighborhoods, these common neighborhoods might have huge overlap (it's possible that all these Δ-tuples have nearly the same common neighborhood), in which case you can't do better than a greedy embedding. But this idea allows us to do much better if we know something about the local connectivity of H.



§2.3 The seeded DRC algorithm

Here's a sketch of the seeded DRC algorithm. We start with $H = (V_H, F_H, E_H)$ and $G = (V_G, F_G, E_G)$, and our goal is to embed H into G.

We'll have a loop, and within this loop, there are three procedures:

(1) First, we embed the next portion of *seeds* (these are abstract points, not related to H) into V_G.

(2) Next, we map the next portion of vertices of F_H into the common neighborhoods of certain subsets of seeds (we use both the newly embedded seeds and seeds from previous steps).

(3) Then we embed the next portion of V_H into V_G, in a way that preserves adjacency.

So now we're simultaneously embedding two types of objects on the left — left vertices of H as well as seeds. (In our realization of seeded DRC, we embed K seeds on step 0, and 1 seed on each step after; but these numbers can vary.)

So we embed a few seeds, and then take the common neighborhoods of these seeds and the seeds embedded previously (or rather, subsets of the seeds).

Then we embed the next portion of the right vertices of H into this common neighborhood. And then we take some left vertices of H which are adjacent to this portion of right vertices, and embed them into V_G in such a way that preserves adjacency.

The advantage over standard DRC is that if we define things correctly, then we introduce additional independence. If we compare the embeddings of the first portion of F_H and some portion of F_H in the middle or end, then these embeddings are going to happen into almost independent sets of F_G — we're embedding into the common neighborhoods of some randomly selected seeds, which are going to be essentially independent. And this means if we consider the common neighborhoods of our embedded points, those will also be essentially independent, so they'll have very small overlap. So this overcomes the main issue of standard DRC — the overlap of common neighborhoods, which prevents you from improving over the greedy embedding procedure.

Here we're reviewing seeds online and embedding, rather than taking all the seeds and then embedding everything (as in standard DRC). We need some rules for how we embed things — in particular, the choice of seeds can depend on how we embedded the left vertices on previous steps. Very roughly, what we want to ensure is that in our embedding, the common neighborhoods of seeds or vertices should be large, and they shouldn't intersect too much with the ones we embedded at previous steps. So we need some rule for how to embed new portions that ensures this. (As we go, we'll keep track of bad and good sets — sets that we need to avoid or use.)

Remark 2.1. This sort of DRC in a different form (with seeded constructions on *both* sides) was used by Lee to get (1.2), with a bandwidth assumption of $\beta \le \exp(-C\sqrt{\Delta \log n})$. (There's a gap between this and (1.1) if Δ grows slowly with n, and Theorem 1.14 removes most of this gap.)

§2.4 Ordering vertices and seeds

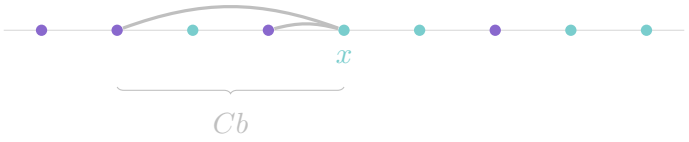
Now we'll talk a bit about some of the technical details. First, there's the ordering. We have three types of objects we're trying to embed — seeds, left vertices, and right vertices — and we need to define the order in which we embed them.

First we'll order the vertices. We can show that if H has bandwidth b, then up to increasing the bandwidth parameter by a constant factor, we can get an arrangement of vertices — still satisfying Definition 1.10 — such that every left vertex comes after all right vertices adjacent to it.

Proposition 2.2

If H is bipartite and has bandwidth at most b, then we can find an ordering such that:

- Any two neighbors are at most Cb apart.
- For every $x \in V_H$, all vertices in F_H which are adjacent to x come before x in the ordering.

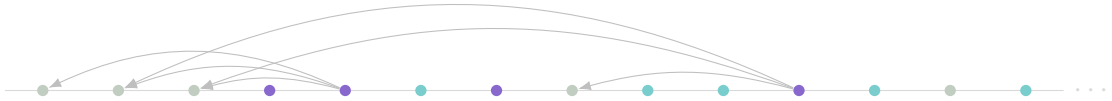


This defines the order in which we embed vertices — so we're embedding left vertices only after all the relevant right vertices have already been embedded.

Now we need to order the seeds. We'll have some parameter K which controls the number of seeds per right vertex — every right vertex will correspond to some K-subset of seeds. On the first step, we start by embedding K seeds. Then we jump over roughly Cb vertices and insert another seed; then we again jump over Cb vertices and insert one more seed; and we continue doing this until we reach the end of the sequence. This means the total number of seeds is

$$L \approx K + \frac{|V_H \cup F_H|}{Cb}.$$

Then for each right vertex, we assign to it the K latest seeds (i.e., the K seeds which precede it).



Then we embed seeds and vertices according to whatever sequence we get, one by one — we look at the next element of the ordering; if it's a seed we embed it on the left, if it's a left vertex of H then we embed it on the left, and if it's a right vertex of H then we embed it on the right.

As we embed, we need to keep track of certain properties — for example, we want to require that the number of common neighbors for subsets of seeds remain large. So we'll have lots of constraints and we need to verify that they can be satisfied at each step; this is the challenging part of the proof.

§2.5 Preprocessing

We won't go over the details of the preprocessing step, but we'll roughly talk about the properties of the preprocessed subgraph G.

The main property we want is control over common neighborhoods — we need the number of common neighbors of small tuples on the left to be large. And we can satisfy this because the graph Γ we start with has constant edge density. So we can simply remove vertices of low degree or small tuples of vertices whose common neighborhoods are tiny; and by doing this iteratively, we can ensure that all h-tuples of left vertices have at least $(c\alpha)^h |F_G|$ common neighbors.

(The rest of the proof is pretty technical, and is about keeping track of various cardinalities of bad and admissible sets.)

§3 Ramsey numbers of geometric graphs

Finally, we'll talk about an open question. Theorem 1.14 doesn't help us understand what happens with Ramsey numbers of *geometric* graphs. Here's an example of a geometric graph:

Example 3.1

Consider the graph in $\mathbb{R}^k$ obtained by taking n independent standard Gaussians in $\mathbb{R}^k$, and drawing an edge between two vectors if their dot product is at least some threshold. (By setting this threshold, we can control the degrees.)

It's known that if $k \gg \log n$, then the resulting geometric graph will be very similar to the Erdős–Rényi random graph of the same average degree — for example, it'll be a strong expander. In particular, it's likely that its Ramsey number will have the same behavior that we saw for random graphs of maximum degree Δ — it'll depend exponentially in the average degree.

Question 3.2. When does the Ramsey number of such a graph become linear in n?

It's not clear, but it seems plausible that if $k = o(\log n)$ and you choose a threshold such that the geometric graph has average degree $\Delta = o(\log n)$, then the Ramsey number might be linear. (We can imagine inserting some factors of $\log \log n$ in these bounds.) This seems reasonable to expect, but Theorem 1.14 doesn't capture this because bandwidth is the wrong quantity to consider for such graphs — they can't be ordered efficiently as in the definition of bandwidth.