FINDING PARTITE HYPERGRAPHS EFFICIENTLY

FERRAN ESPUÑA

Abstract. We give a deterministic polynomial-time algorithm that, for a given k-uniform hypergraph H with n vertices and edge density d, finds a $K(t, \overset{k}{\ldots}, t)$ subgraph with parts of size at least $c_d(\log n)^{1/(k-1)}$, building on work by Mubayi and Turán for the $k = 2$ case. This value for the part size matches the order of magnitude guaranteed by the non-constructive proof due to Erdős and is tight up to a constant factor.

1. Introduction

Hypergraph Turán problems study how many edges a k-uniform hypergraph $H = (V, E)$ with n vertices can have without containing a specific subgraph G. The maximal such number is known as the *Turán number* $\operatorname{ex}(n, G)$. It is known [2] that $\operatorname{ex}(n, G) = o\left(\binom{n}{k}\right)$ if and only if G is k-partite, i.e., if its vertex set can be partitioned into k disjoint sets such that each edge contains exactly one vertex from each part. Kővári, Sós, and Turán [3] (for $k = 2$) and Erdős [1] (for any $k \geq 2$) established that

$$\operatorname{ex}(n, K(t, \overset{k}{\ldots}, t)) = \mathcal{O}\left(n^{k - \frac{1}{t^{(k-1)}}}\right),$$

where $K(t, \overset{k}{\ldots}, t)$ is the complete balanced k-partite k-graph with k parts of size t. Furthermore, if H is a k-graph with at least $d\binom{n}{k}$ edges for some constant $d > 0$, then it contains a $K(t, \overset{k}{\ldots}, t)$ with $t = c_d \log(n)^{1/(k-1)}$.

This result is non-constructive, meaning it guarantees the existence of such a subgraph but does not provide an efficient way to find it. Note that a simple brute-force search for a $K(t, \overset{k}{\ldots}, t)$ would involve checking all $\binom{n}{kt}$ vertex subsets, which is superpolynomial in n for $t = \Theta((\log n)^{1/(k-1)})$. Mubayi and Turán [4] developed a polynomial-time algorithm for the case $k = 2$, which reaches the stated order of magnitude for the subgraph part size. This work extends their approach to the general case of k-uniform hypergraphs, reaching analogous results for $k \geq 3$. More concretely, we prove the following.

Theorem 1.1. *There is a deterministic algorithm that, given a k-graph H with n vertices and $m = d\binom{n}{k}$ edges, finds a complete balanced k-partite subgraph $K(t, \overset{k}{\ldots}, t)$ in polynomial time, where*

$$t = t(n, d, k) = \left\lfloor \left(\frac{\log n}{\log(16/d)}\right)^{\frac{1}{k-1}} \right\rfloor$$

This value of t matches the order of magnitude from existence proofs. In fact, a probabilistic argument shows that it is the best possible up to a constant factor.

2. The algorithm

We present a recursive algorithm, `FindPartite`, that finds a $K(t, \overset{k}{\ldots}, t)$ in a given k-graph H. The core idea is to reduce the uniformity of the problem from k to $k - 1$ in each recursive step. The algorithm takes a k-graph H with n vertices and m edges as input. It first defines the target part size t, a small set size w, and a threshold edge count s for the recursive call, based on the input graph's parameters:

$$
\begin{aligned}
t = t(n,d,k) &= \left\lfloor \left(\frac{\log n}{\log(16/d)} \right)^{\frac{1}{k-1}} \right\rfloor, \\
w = w(n,d,k) &= \left\lceil \frac{4t}{d} \right\rceil, \text{ and} \\
s = s(n,d,k) &= \left\lceil \left(\frac{d}{4}\right)^{t} \binom{n}{k-1} \right\rceil,
\end{aligned}
$$

where $d = \frac{m}{\binom{n}{k}}$ is the edge density of H. The main steps are:

(1) **Base Case ($k = 1$):** The edge set of a 1-graph is just a collection of vertices. Return the set of all vertices that are "edges".

(2) **Select High-Degree Vertices:** Choose a set $W \subset V$ of w vertices with the highest degrees in H.

(3) **Find a Dense Link Graph:** Iterate through all t-subsets $T \subset W$. For each T, consider the set S of all $(k-1)$-subsets of V that form a hyperedge with *every* vertex in T.

(4) **Recurse:** As we prove further along using the Kővári–Sós–Turán theorem, for at least one choice of T, the resulting set S will be large ($|S| \geq s$). We form a new $(k-1)$-graph $H' = (V, S)$ and make a recursive call: `FindPartite(`H'`, `$k-1$`)`.

(5) **Construct Solution:** The recursive call returns $k-1$ parts $V_1, \dots, V_{k-1}$ of size at least t. By construction, every choice of vertices from these parts forms an edge in H' with every vertex of T. Thus, $(T_1, \dots, T_{k-1}, T)$ form the desired $K(t, \overset{k}{\dots}, t)$ in the original graph H.

The pseudocode is given in Algorithm 1.

Algorithm 1 Finding a balanced partite k-graph

function FINDPARTITE(H, k)
  if $k = 1$ **then**
    return $(\{x \colon \{x\} \in E(H)\})$
  end if
  $n \leftarrow |V(H)|$, $m \leftarrow |E(H)|$, $d \leftarrow \frac{m}{\binom{n}{k}}$
  $t \leftarrow t(n,d,k)$, $w \leftarrow w(n,d,k)$, $s \leftarrow s(n,d,k)$
  assert $t \geq 2$
  $W \leftarrow$ a set of w vertices with highest degree in H
  for all $T \in \binom{W}{t}$ **do**
    $S \leftarrow \{\, y \in \binom{V}{k-1} \colon \forall x \in T, \{x\} \cup y \in E(H) \,\}$
    if $|S| \geq s$ **then**
      $H' \leftarrow (V, S)$ ▷ H' is a $(k-1)$-graph
      $(V_1, \dots, V_{k-1}) \leftarrow$ FINDPARTITE$(H', k-1)$
      return $(V_1, \dots, V_{k-1}, T)$
    end if
  end for
end function

We now present the proof of correctness and polynomial runtime for our algorithm. We assume $t \geq 2$ for our estimates to be easier. If $t < 2$, we may just return the vertices of any single edge in H.

3. Proof of correctness

It is not immediately clear that the set W defined in step 2 of the algorithm is well-defined, as for this it is necessary that $w \leq n$. To show this, we first observe that our assumption $t \geq 2$ implies that $1 \geq d \geq \frac{16}{\sqrt{n}}$.

Suppose, by way of contradiction, that $w > n$. Then, we have

$$n \le w - 1 \le \frac{4t}{d} \le \frac{4\log n}{d\log(16/d)} \le \frac{4\sqrt{n}\log n}{16\log(16/d)}.$$

Taking, for example, the logarithms to be in base e, we note that $\log x \le \sqrt{x}$ for all positive x, and that $\log(16/d) \ge \log(16) > 1$. Therefore, we get $n \le \frac{n}{4}$, which is a contradiction.

Next, we will prove that in step 3 of the algorithm we indeed find a set $T \in \binom{W}{t}$ such that the associated set $S \subset \binom{V}{k-1}$ has size at least s. That is, Algorithm 1 reaches line 13 at some point in the for loop. For this, consider the bipartite graph B with parts $\binom{V}{k-1}$ and W with edge set

$$\left\{(x,y) \in \binom{V}{k-1} \times W \,\middle|\, x \cup \{y\} \in E\right\}.$$

The edges of B correspond to the edges containing each vertex in W, so there are

$$z = \sum_{y \in W} d_H(y) \ge k \cdot m \cdot \frac{w}{n} = \frac{k \cdot w \cdot d \cdot \binom{n}{k}}{n} = w \cdot d \cdot \binom{n-1}{k-1}$$

of them, where the inequality follows from the fact that we have picked a set of w vertices with highest degree in H. The existence of a set $T \subset W$ as desired is equivalent to there being $T \subset W$ of size t and a set $S \subset \binom{V}{k-1}$ of size s such that the induced bipartite subgraph $B[S,T]$ is complete. To prove that this is the case, we use a version the Kővári–Sós–Turán theorem [3], which we state and prove here for completeness.

Lemma 3.1. *Let u, w, s, t be positive integers with $u \ge s$, $w \ge t$, and let B be a bipartite graph with parts W and U such that $|U| = u, |W| = w$. If B has more than*

$$(s-1)^{1/t}(w-t+1)u^{1-1/t} + (t-1)u$$

edges, then there are $T \subset W$ of size t and $S \subset U$ of size s such that the induced bipartite subgraph $B[T,S]$ is complete.

We apply this lemma with $u = \binom{n}{k-1}$. It is clear from the definitions that our parameter satisfy the requirements $u \ge s$ and $w \ge t$. Suppose, by way of contradiction, that

$$w \cdot d \cdot \binom{n-1}{k-1} \le (s-1)^{1/t}(w-t+1)\binom{n}{k-1}^{1-1/t} + (t-1)\binom{n}{k-1}.$$

Algebraic manipulation then shows that

$$\frac{1}{2} \cdot w \cdot d \le w \cdot d \cdot \left(1 - \frac{k}{n}\right) \le w\left(\frac{s-1}{\binom{n}{k-1}}\right)^{1/t} + (t-1),$$

where the first inequality follows from $n \ge 2k$, which follows from $t \ge 2$. Finally, since $t \le \frac{w \cdot d}{4}$ by the definition of w, we obtain

$$\left(\frac{d}{4}\right)^t \binom{n}{k-1} < s - 1,$$

against the definition of s. We are now ready to prove that the algorithm returns a $K(t, \overset{k}{\ldots}, t)$. More precisely, we show the following.

Theorem 3.2. *For $k \ge 2$, if $t \ge 2$, Algorithm 1 returns a tuple $(V_1, \ldots, V_k)$ of disjoint sets $V_i \subset V(H)$ such that $|V_i| \ge t$ and $H[V_1, \ldots, V_k]$ is complete.*

Proof. We proceed by induction on k. For $k = 2$, the recursive call returns the common neighborhood V_1 of the vertices in T, which is obviously disjoint from T, so it only remains to check that $|V_1| \ge t$. Now, since by construction $|V_1| = |S| \ge s$, it is enough that

$$s = \left\lceil \left(\frac{d}{4}\right)^t n \right\rceil \ge \left(\frac{d}{4}\right)^{\frac{\log n}{\log(16/d)}} n = \frac{1}{n} \cdot 4^{\frac{\log n}{\log(16/d)}} \cdot n \ge 4^t > t.$$

For $k \geq 3$, we assume the inductive hypothesis holds for $k-1$. If d' is the edge density of the $(k-1)$-graph H' and $t' = t(n, d', k-1)$, as long as $t' \geq 2$, the recursive call returns a tuple $(V_1, \ldots, V_{k-1})$ of disjoint sets $V_i \subset V(H)$ such that $|V_i| \geq t'$ and $H'[V_1, \ldots, V_{k-1}]$ is complete.

We claim that $t' \geq t$. This implies that $t' \geq 2$ so we get to apply the inductive hypothesis to H'. Furthermore, the sets V_i are at least as large as desired. By construction of H', for all $(x_1, \ldots x_{k-1}, y) \in V_1 \times \cdots \times V_{k-1} \times T$, we have that $\{x_1, \ldots, x_{k-1}, y\} \in E(H)$. In particular, because all V_i are nonempty, this implies that T is disjoint from each of them. Furthermore, $H[V_1, \ldots, V_{k-1}, T]$ is complete, finishing the proof.

Let us now prove the claim that $t' \geq t$. By the definition of s, we have $d' \geq \left(\frac{d}{4}\right)^t$. Therefore,

$$t' \geq \left\lfloor \left(\frac{\log n}{\log\left(\frac{16}{(d/4)^t}\right)} \right)^{\frac{1}{k-2}} \right\rfloor \geq \left\lfloor \left(\frac{\log n}{\log 16 - t\log(d/4)} \right)^{\frac{1}{k-2}} \right\rfloor.$$

Then, we substitute the definition of t, where removing the floor function maintains the inequality because the right hand side is decreasing in t (recall $d \leq 1$):

$$t' \geq \left\lfloor \left(\frac{\log n}{\log 16 - \left(\frac{\log n}{\log(16/d)}\right)^{\frac{1}{k-1}} \log(d/4)} \right)^{\frac{1}{k-2}} \right\rfloor = \left\lfloor \left(\frac{(\log n)^{\left(1-\frac{1}{k-1}\right)}}{\frac{\log 16}{(\log n)^{\frac{1}{k-1}}} - \frac{\log(d/4)}{\log(16/d)^{\frac{1}{k-1}}}} \right)^{\frac{1}{k-2}} \right\rfloor.$$

If we bound the denominator by showing that

$$\frac{\log 16}{(\log n)^{\frac{1}{k-1}}} - \frac{\log(d/4)}{\log(16/d)^{\frac{1}{k-1}}} \leq (\log(16/d))^{\left(1-\frac{1}{k-1}\right)}, \tag{1}$$

then the expression simplifies to

$$t' \geq \left\lfloor \left(\frac{(\log n)^{\left(1-\frac{1}{k-1}\right)}}{(\log(16/d))^{\left(1-\frac{1}{k-1}\right)}} \right)^{\frac{1}{k-2}} \right\rfloor = \left\lfloor \left(\frac{\log n}{\log(16/d)} \right)^{\frac{1}{k-2}\left(1-\frac{1}{k-1}\right)} \right\rfloor = \left\lfloor \left(\frac{\log n}{\log(16/d)} \right)^{\frac{1}{k-1}} \right\rfloor = t,$$

as desired. Suppose, by way of contradiction, that Inequality (1) does not hold. We can rewrite

$$(\log(16/d))^{\left(1-\frac{1}{k-1}\right)} = \frac{\log(16/d)}{\log(16/d)^{\frac{1}{k-1}}}$$

and rearrange the inequality to obtain

$$\frac{\log 16}{(\log n)^{\frac{1}{k-1}}} > \frac{\log(16/d) + \log(d/4)}{\log(16/d)^{\frac{1}{k-1}}} = \frac{\log 4}{(\log(16/d))^{\frac{1}{k-1}}}.$$

This implies that

$$t \leq \left(\frac{\log n}{\log(16/d)} \right)^{\frac{1}{k-1}} < \frac{\log 16}{\log 4} = 2,$$

which contradicts the assumption that $t \geq 2$. □

4. Proof of Polynomial Complexity

We now analyze the computational complexity of Algorithm 1 to show that it runs in time polynomial in n for any fixed uniformity k.

Let $T_k(n)$ denote the worst-case running time of the function `FindPartite` when called on a k-uniform hypergraph with n vertices. The algorithm's structure gives a recurrence relation for $T_k(n)$. We will analyze the cost of the operations within a single call, excluding the recursive step.

First, consider the initialization steps. The parameters n, m, d, t, w, s are computed. This requires knowing the number of vertices and edges. The number of edges m can be counted in $O(n^k)$ time, but it is often given as part of the input. Computing the degrees of all vertices requires iterating through all m edges, taking $O(k \cdot m) = O(k \cdot d\binom{n}{k}) = O(n^k)$ time. Once degrees are known, the set W of the w vertices with the highest degrees can be found in $O(n)$ time using a selection algorithm (e.g., Quickselect). Thus, the cost of all steps before the main loop is $O(n^k)$.

The main part of the computation happens in the `for` loop, which iterates through all t-subsets T of W. The number of iterations is bounded by $\binom{w}{t}$. As shown in the proof of correctness (and can be verified with a more detailed calculation), this quantity is bounded by a polynomial in n. Specifically, using the bound $t \geq 2$:

$$\binom{w}{t} \leq \left(\frac{ew}{t}\right)^t \leq \left(\frac{e(4t/d+1)}{t}\right)^t = \left(\frac{4e}{d} + \frac{e}{t}\right)^t \leq \left(\frac{5e}{d}\right)^t$$

Since $t = \Theta((\log n)^{1/(k-1)})$ and d is bounded below by a term related to $n^{-1/2}$, it follows that $\log(\binom{w}{t}) = O(t \log(1/d)) = O(\log n)$. This implies that $\binom{w}{t} = n^{O(1)}$, i.e., the number of iterations is polynomially bounded in n. Let's denote this bound by $\mathrm{poly}(n)$.

Inside each iteration of the loop, the algorithm constructs the set S. A straightforward way to implement this is to iterate through all $\binom{n}{k-1} = O(n^{k-1})$ possible $(k-1)$-subsets $y \subset V$. For each y, we must check if $\{x\} \cup y$ is an edge in H for all $x \in T$. Assuming the edge set $E(H)$ is stored in a hash table or a similar structure that allows for constant-time lookups on average, each check takes $O(k)$ time. The check is performed for all t vertices in T. Therefore, the cost of constructing S in a single iteration is:

$$O\left(\binom{n}{k-1} \cdot t \cdot k\right) = O(n^{k-1} \cdot t)$$

The algorithm makes at most one recursive call, as it returns immediately after a successful call. Therefore, the total running time $T_k(n)$ is governed by the recurrence:

$$T_k(n) = O(n^k) + \mathrm{poly}(n) \cdot O(t \cdot n^{k-1}) + T_{k-1}(n)$$

The term for the loop dominates the $O(n^k)$ initialization cost. Let's denote the non-recursive work at level k by $P_k(n) = \mathrm{poly}(n) \cdot O(t \cdot n^{k-1})$. Since t grows polylogarithmically with n, $P_k(n)$ is a polynomial in n for any fixed k.

The recurrence relation simplifies to:

$$T_k(n) = P_k(n) + T_{k-1}(n)$$

with a base case of $T_1(n) = O(n)$. Unrolling this recurrence, we get:

$$T_k(n) = P_k(n) + P_{k-1}(n) + \cdots + P_2(n) + T_1(n)$$

This is a sum of $k-1$ polynomials in n. The total running time is therefore dominated by the polynomial of the highest degree, which is $P_k(n)$.

$$T_k(n) = O(P_k(n)) = \mathrm{poly}(n) \cdot O(t \cdot n^{k-1})$$

As this expression is a polynomial in n for any fixed k, we conclude that the algorithm runs in polynomial time.

5. Conclusion and Future Work

We have presented a deterministic, polynomial-time algorithm to find a large complete balanced k-partite subgraph in any sufficiently dense k-uniform hypergraph. This provides a constructive counterpart to a classical existence result by Erdős in extremal hypergraph theory.

Several avenues for future research remain open.

- **General Blow-ups:** Our algorithm finds a blow-up of a single edge, $K(t, \overset{k}{\ldots}, t)$. Can this framework be adapted to find a t_n-blowup of an arbitrary fixed k-graph G? Existence theorems guarantee such structures, but efficient algorithms are lacking.
- **Unbalanced Partite Graphs:** The algorithm could be modified to search for unbalanced complete partite graphs $K(t_1, \ldots, t_k)$, where the part sizes may grow at different rates.
- **Optimality:** The bounds on t are asymptotically tight, but the constants can likely be improved with a more refined analysis. For $k = 2$, it is known that in dense graphs one can find a $t = \Theta(\log n)$ blow-up of any bipartite graph. It is an open question if a constructive proof for this stronger result exists for $k \geq 2$.

Acknowledgements. The ideas in this work stem from the author's master's thesis of the same name, Universitat Politècnica de Catalunya, under the supervision of Richard Lang. The author thanks Dr. Lang for suggesting the problem and for his guidance and support throughout the project.

References

[1] Paul Erdős. On extremal problems of graphs and generalized graphs. *Israel Journal of Mathematics*, 2:183–190, 1964.
[2] Peter Keevash. Hypergraph Turán problems. In *Surveys in combinatorics 2011*, volume 392 of *London Math. Soc. Lecture Note Ser.*, pages 83–139. Cambridge Univ. Press, Cambridge, 2011.
[3] Thomas Kövari, Vera T. Sós, and Pál Turán. On a problem of K. Zarankiewicz. *Colloquium Mathematicum*, 3:50–57, 1954.
[4] Dhruv Mubayi and György Turán. Finding bipartite subgraphs efficiently. 110(5):174–177, 2010.