Artificial Intelligence and Unit Distance Graphs

Let $X \subseteq \mathbf{R}^n$ be a finite subset. We can give X the structure of an undirected graph called a *Unit Distance Graph (UDG)* by letting the edge set comprise those vertex pairs $\{x, y\} \subseteq X$ that are unit distance apart: $|x - y| = 1$. The question of what is the maximum number of edges of a UDG of a given number of vertices is still open since was posed by Erdős in 1946 [1]. Even the asymptotic behavior is unclear as the known upper and lower bound are quite far apart: see [2] resp. [3] for the planar, and [4] resp. [5] for the spatial case.

The aim of this project is to study UDGs that are *dense*, that is they have a high edge count. We try to use computer search. This is an ongoing project, involving several BSM students and multiple members of the AI group at the Rényi Institute.

With the Spring and Summer 2023 groups, we developed a computer search algorithm that could find all the densest known planar UDGs in [2, Table 1], and moreover go on and find dense UDGs up to vertex number 100. We recently finished the paper that describes our method [6].

With the groups that came after, we were working on multiple directions to improve the result and extend it to other Euclidean spaces like the 2-sphere or 3-dimensional space. I will be happy to expound in person on the many great ideas that recent groups came up with and explored.

This semester, our primary focus will be realization: given an abstract graph, can we decide if it is isomorphic to a UDG in a given Euclidean space, or yet better, can we enumerate all possible realizations, if they exist?

Prerequisites

Strong command of the `Python` numerical library `numpy`.

Qualifying problems

1. Write a function

```
get_new_vertices(
    addends: np.ndarray,
    parent: np.ndarray
) -> np.ndarray
```

that given an integer array `addends` of shape (a, d) and an integer array `parent` of shape (n, d), returns an array `new_vertices` of shape $(a \cdot n, d)$ the rows of which are $u + v$ such that u is a row of `addends` and v is a row of `parent`.

Bonus: Make the function accept batches of parents. That is, if `parent` has shape $(b_1, \ldots, b_k, n, d)$, then the function should return an array `new_vertices` of shape $(b_1, \ldots, b_k, a \cdot n, d)$ such that for $0 \leq i_j < b_j$ for $1 \leq j \leq k$, the 2-array `new_vertices[`$i_1, \ldots, i_k$`]` is equal to `get_new_vertices(addends, parent[`$i_1, \ldots, i_k$`])`.

2. Write a function

```
get_dedupe_mask(
    new_vertices: np.ndarray,
    parent: np.ndarray
) -> np.ndarray
```

that given an integer array `new_vertices` of shape (m, d) and an integer array `parent` of shape (n, d), returns a Boolean array `dedupe_mask` of shape $(m,)$ such that for $0 \leq i < m$, the Boolean value `dedupe_mask[`i`]` says if `new_vertices[`i`]` is not equal to a row of `parent` or a row of `new_vertices[:`i`]`.

Bonus: Make the function accept batches of new vertices and parents. That is, if `new_vertices` has shape $(b_1, \ldots, b_k, m, d)$ and `parent` has shape $(b_1, \ldots, b_k, n, d)$, then the function should return a Boolean array `dedupe_mask` of shape $(b_1, \ldots, b_k, m)$ such that for $0 \le i_j < b_j$ for $1 \le j \le k$, the 1-array `dedupe_mask[`$i_1, \ldots, i_k$`]` is equal to `get_dedupe_mask(new_vertices[`$i_1, \ldots, i_k$`], parent[`$i_1, \ldots, i_k$`])`.

In each problem, to improve your evaluation, you can try to *vectorize* your solution, that is use as few explicit iterations (such as ones using `for`, `while`, `map`, recursion, etc.) along array dimensions as possible. This will greatly improve the speed of your code. To this end, you can peruse the built-in functions in `numpy`, advanced indexing and broadcasting.

You can hand in the qualifying problems by writing the functions in the file `qualifying_problems_fall_2024.py` included in the research project description. The file includes test cases, you just have to run it to test your solutions.

You can hand in multiple versions. Of each problem, I will evaluate the latest submission. A valid submission must arrive to my email address by **August 27, 11:59PM CEST (UTC +02:00)**. You can expect me to answer a question before this time if it arrived to my email address by **August 26, 11:59PM CEST (UTC +02:00)**.

In your email, please also write me the following:

1. Your Mathematics and Computer Science background.
2. Your Mathematics and Computer Science interests.
3. What do you find especially interesting in this project?

Have fun with the problems and hope to see you in the group!

Contact

Pál Zsámboki, `zsamboki@renyi.hu`, Alfréd Rényi Institute of Mathematics

References

[1] Pál Erdős. On sets of distances of n points. *The American Mathematical Monthly*, 53(5):248–250, 1946. ISSN 00029890, 19300972. URL http://www.jstor.org/stable/2305092.

[2] Péter Ágoston and Dömötör Pálvölgyi. An improved constant factor for the unit distance problem. *Studia Scientiarum Mathematicarum Hungarica*, 59(1):40 – 57, 2022. doi: https://doi.org/10.1556/012.2022.01517. URL https://akjournals.com/view/journals/012/59/1/article-p40.xml.

[3] Pál Erdős. *Some of my favourite unsolved problems*, page 467–478. Cambridge University Press, 1990. doi: 10.1017/CBO9780511983917.039.

[4] Haim Kaplan, Jiří Matoušek, Zuzana Safernová, and Micha Sharir. Unit distances in three dimensions. *Combinatorics, Probability and Computing*, 21(4):597–610, 2012. doi: 10.1017/S0963548312000144.

[5] Pál Erdős. On sets of distances of n points in euclidean space. *Magyar Tudományos Akadémia Matematikai Kutatóintézet Közleményei*, 5:165–169, 1960. URL http://www.math-inst.hu/~p_erdos/1960-08.pdf.

[6] Peter Engel, Owen Hammond-Lee, Yiheng Su, Dániel Varga, and Pál Zsámboki. Diverse beam search to find densest-known planar unit distance graphs, 2024. URL https://arxiv.org/abs/2406.15317. To be submitted.