Artificial Intelligence and Unit Distance Graphs

Let $X \subseteq \mathbf{R}^2$ be a finite planar subset. We can give X the structure of an undirected planar graph called a *Unit Distance Graph* by letting the edge set comprise those vertex pairs $\{x, y\} \in X$ 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 [2] and lower [3] bounds are quite far apart.

There are at least two research directions one could go with this, that quite probably involve the use of computer search.

1. In Spring and Summer 2023 RES projects, we have developed a computer search algorithm that could find all the best known UDGs in [2, Table 1] and moreover go on and find dense UDGs up to vertex number 100. We can probably go up to a few hundred vertices. We seek to improve the lower bound by trying to find in this database a construction pattern that we can continue indefinitely.
2. Also, the question of what is the densest UDG can be similarly studied for UDGs on the sphere $X \subseteq S^2$, in the space $X \subseteq \mathbf{R}^3$, etc. Here, we could again try to look for dense UDGs by computer search.

Prerequisites

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

Qualifying problem

The `Moser lattice` is the lattice

$$\{a + b\omega_2 + c\omega_3 + d\omega_2\omega_3 : a, b, c, d \in \mathbb{Z}\} \subset \mathbb{C} \text{ where } \omega_2 = \frac{1}{2} + i\frac{\sqrt{3}}{2} \text{ and } \omega_3 = \frac{5}{6} + i\frac{\sqrt{11}}{6}.$$

If the points of a UDG of n vertices are in the Moser lattice, then it can be given by the $(n, 4)$ matrix of coefficients in the base $(1, \omega_2, \omega_3, \omega_2\omega_3)$. Write a function

```
rotate(coefficients: ndarray) -> ndarray
```

that given a UDG with Moser lattice coefficients outputs the Moser lattice coordinates of the UDG that has been rotated by $\frac{\pi}{3}$ counterclockwise around the origin. For example, here's how the Moser spindle would rotate:

```
assert(np.array_equal(
    rotate(np.array([[0,0,0,0], [1,0,0,0], [0,1,0,0], [1,1,0,0], [0,0,1,0], [0,0,0,1], [0,0,1,1]])),
    np.array([[0,0,0,0], [0,1,0,0], [-1,1,0,0], [-1,2,0,0], [0,0,0,1], [0,0,-1,1], [0,0,-1,2]
    ]
)
```

Try not to use a `python` loop but use effective, vectorized `numpy` operations.

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.