CrystEngComm

PAPER

View Article Online
View Journal | View Issue

Cite this: *CrystEngComm*, 2024, **26**, 192

Unravelling the structure of the CSD cocrystal network using a fast near-optimal bipartisation algorithm for large networks†

Tom E. de Vries, Elias Vlieg and René de Gelder *

Networks, consisting of vertices connected by edges, are an important mathematical concept used to describe relationships between people, roads between cities, reactions between chemicals, and many other interactions. Such a network can be created by extracting cocrystals from the Cambridge Structural Database (CSD). This network describes which compounds can form cocrystals together and can, for example, be used to predict new cocrystals using link-prediction techniques. Bipartiteness is an important property of some networks wherein the vertices can be separated into two groups such that edges only point from one group to the other. Knowing whether a network is bipartite can make studying its structure considerably easier. If a network is nearly bipartite except for a number of outlying edges, one might want to identify and remove those edges, thereby bipartising the network. The CSD cocrystal network was previously found to be close to bipartiteness. Truly bipartising it could improve the accuracy of link-prediction and give insight into the hidden structure of the network. Many algorithms exist for exactly finding the optimal bipartisation for a nearly-bipartite network, but the time it takes to complete such algorithms increases exponentially with the size of the problem. In some cases, an exact solution is unnecessary and a 'good enough' bipartisation is sufficient. We have developed an algorithm that can find a near-optimal bipartisation within reasonable time, even for very large networks, and used it to unravel the structure of the CSD cocrystal network. We obtained a bipartisation that leaves 96% of the network intact, and we were able to identify 'universal' coformers that do not conform to the bipartite nature of the network. By applying a clustering algorithm to the bipartised network, we were also able to identify anticommunities of coformers.

Received 6th October 2023,
Accepted 23rd November 2023

DOI: 10.1039/d3ce00978e

rsc.li/crystengcomm

1 Introduction

Networks are mathematical constructs that can be used to describe the relations between different entities. Common examples include networks that describe roads between cities, gene interactions, and social networks.[1,2] A network is considered bipartite if you can split it into two parts, such that there are only connections from one part to the other, and no connections inside each part. Bipartiteness is an important property that can be exploited in network science methods like link-prediction.[3] Determining whether or not a network is bipartite is a well-studied problem, along with the equivalent two-colouring problem.[4] A more complex extension of this problem is that of bipartisation, also known as odd-cycle transversal (OCT): starting with a non-bipartite network and turning it into a bipartite network while changing as little as possible.

The network that we are particularly interested in here is the one created from the list of known cocrystals extracted from the Cambridge Structural Database (CSD). This network represents a large number of co-crystallising compounds called coformers, and describes which pairs of these coformers can form cocrystals together. Devogelaer *et al.*[5] have examined the CSD cocrystal network by looking at its degree distribution and applying clustering algorithms. By locally counting bipartite and monopartite paths it was shown that 99% of the cocrystal network behaves in a bipartite manner. Other works have claimed that the network is monopartite, but base this assertion solely on the fact that it is not perfectly bipartite.[6] The method used by Devogelaer *et al.* to determine how close the network is to true bipartiteness is somewhat heuristic, and there are still questions left unanswered: can we use a bipartisation algorithm to verify this percentage in a more exact way? Is the bipartite behaviour found by Devogelaer *et al.* a local property of individual links, or is it a global property of the network as a whole? Can we use our bipartisation algorithm to identify those coformers that do not fit the bipartite behaviour of the rest of the network and behave in a monopartite or universal manner? And lastly, can we use our algorithm to more clearly visualise

Solid State Chemistry, Institute for Molecules and Materials, Radboud University, Heyendaalseweg 135, 6525 AJ Nijmegen, The Netherlands.
E-mail: r.degelder@science.ru.nl

† Electronic supplementary information (ESI) available. See DOI: https://doi.org/10.1039/d3ce00978e

the structure of the network as a whole and examine it in more detail?

Bipartisation algorithms can be used to see how close a network is to being bipartite by examining how many of its vertices or edges need to be removed. Identifying 'monopartite' elements in a network can help improve link-prediction methods that rely on the bipartite nature of a network by removing those monopartite elements. Bipartisation algorithms have applications in *e.g.* quantum computing, wherein a process called quantum annealing requires the bipartisation of a network that represents an optimisation problem to be solved by the quantum computer.[7]

There are various ways of bipartising a network such as deleting vertices, contracting edges, and deleting edges. All of these have been studied extensively, and exact algorithms that can find the optimal bipartisation have been designed for all of them. However, bipartisation by any of these three methods is an NP-hard problem; *i.e.* the time it takes to complete an algorithm scales exponentially with the size of the problem.[8–10] In practice, this means exact bipartisation algorithms cannot be used for large networks, as the completion time would quickly exceed the age of the universe. Networks used to test exact bipartisation methods contain at most a few hundred vertices.[11] The CSD cocrystal network contains over 8000 vertices, so it is definitely too large for exact bipartisation algorithms. To bipartise large networks, it is necessary to use what we call 'near-optimal' algorithms. Such algorithms are designed to deliver a 'good enough' approximation of the optimal bipartisation that would have been delivered by an exact algorithm. This does mean that the algorithm may remove slightly more of the network than is strictly necessary, but since the alternative is not feasible to begin with, near-optimal algorithms are the only option for the bipartisation of large networks.

We have developed a new near-optimal edge bipartisation algorithm for use in large networks. Our algorithm makes use of link-prediction like scoring functions that grade edges according to how well they fit in a bipartite network (see section 2). Other near-optimal bipartisation algorithms such as that designed by Concas *et al.* use eigenvalue decompositions, which are still quite time consuming.[12] Their method also requires several parameters to be tuned to work correctly, which we would like to avoid. Our method is designed to rapidly break down the network into a bipartite 'seed' network, and then grow a new bipartite network from that seed. Application of our method to the CSD cocrystal network showed that it is at least 96% bipartite, and allowed us to closely examine its structure.

2 Methods

2.1 Important concepts

Here we briefly introduce some important network science concepts.

A network is represented by a graph $G = \{V,E,W\}$ consisting of a set of N vertices $V = \{v_i | i \in [1\ldots N]\}$, a set of M edges $E = \{(e_{j,1},e_{j,2}) | j \in [1\ldots M]\}$, and a set of weights $W = \{w_j | j \in [1\ldots M]\}$. An edge $e_j = (e_{j,1},e_{j,2})$ points from vertex $v_{e_{j,1}}$ to vertex $v_{e_{j,2}}$ and has weight w_j. Each vertex v has a degree $\deg(v)$ equal to the sum of the weights of the edges connected to v. Graphs can be represented by an $N \times N$ adjacency matrix A, defined as:

$$A_{i,j} = \begin{cases} w_k & \exists k : e_k = (i,j) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The cocrystal network is an unweighted and undirected network with no self-loops, which means that all its weights are 1, that if $(i, j) \in E$, then $(j, i) \in E$, and that $A_{i,i} = 0$. Thus, A is a symmetrical $N \times N$ matrix with 1 at the location of two vertices that share an edge, and 0 everywhere else. Because all the weights are equal to 1, the degree of a vertex v is equal to the number of edges connected to v.

A network is defined to be bipartite if its vertex set V can be divided into two non-empty subsets V_1 and V_2 with $V_1 \cap V_2 = \varnothing$ and $V_1 \cup V_2 = V$ such that for every edge $(i, j) \in E$, if $v_i \in V_1$, then $v_j \in V_2$ and *vice versa*. In other words, all edges have one end in V_1 and the other end in V_2.

2.2 Building the CSD cocrystal network

We used the CCDC program ConQuest[13] and the CSD python API to search the CSD (version 5.43; November 2021 + 4 updates).

In order to find all the cocrystals in the CSD we extracted every entry that contains two distinct compounds, is organic, contains no ions, is error-free, and has its three-dimensional coordinates determined (this includes structures determined from powder data). It should be noted that the fact that we are using the CSD with these filters could lead to certain bias in our data as we do not include cocrystals that do not form well-defined 3D structures. Next, we used the classifier algorithm written by Devogelaer *et al.*[5] to separate the entries into cocrystals, solvates, and entries with a gas molecule as part of their structure, which uses lists of commonly used solvents and gasses. A list of CSD refcodes for all resulting cocrystals is available as ESI.† The cocrystal entries are split into their constituent parts (called coformers), and a network is constructed with these coformers as its vertices and the cocrystals they form as the edges. If a coformer has multiple enantiomers or multiple refcodes in the CSD, one of them is taken as the representative for all of them, and their vertices are combined into one. For example, if a coformer X has two enantiomers X_R and X_S, then we create one vertex 'X' that combines all cocrystals containing X_R, X_S, or the racemic mixture X_{RS}. Finally, an adjacency matrix is constructed as described in section 2.1.

2.3 Bipartisation

The algorithm begins by loading the adjacency matrix A for the network that needs to be bipartised, as well as the list of edges E in the adjacency matrix. The algorithm then bipartises the network in three major steps:

Step 1 is to identify every three-cycle in the network and remove the 'least bipartite' edge from each one. An n-cycle is

a circular path with a length of n edges. Three-cycles cannot exist within a bipartite network because there is no way to assign all three vertices to a bipartite set without having two connected vertices in the same set. This is demonstrated in Fig. 1. In the rest of this paper, we will use green squares and blue circles to indicate the two bipartite sets. Vertices that are not (yet) assigned to a particular bipartite set will be denoted by a purple diamond.

We want to remove the 'least bipartite' edge from each three-cycle. To identify the 'least bipartite' edge we define a scoring function that grades edges based on how bipartite they behave. This is done by counting the number of three-cycles and four-cycles that an edge is part of. A four-cycle suggests good bipartite behaviour, while a three-cycle suggests poor bipartite behaviour. Ideally, we would also like to include longer cycles, but this would exponentially increase the time needed to complete the algorithm. Written out, the score for an edge (i, j) in the adjacency matrix A is:

$$\mathrm{Score}(i,j) = \sum_{(n,m)\in E/(i,j)} \left(A_{i,n}A_{j,m} + A_{i,m}A_{j,n}\right) - \sum_{k=1}^{N} \left(A_{i,k}A_{j,k}\right). \quad (2)$$

The first term in eqn (2) counts the number of four-cycles that the edge (i, j) is part of. This is done by finding any other edge (n, m) and checking whether n shares an edge with i and m shares an edge with j, or *vice versa*. The second term counts the number of three-cycles that the edge (i, j) is a part of by checking for the existence of a vertex k that shares an edge with both i and j. This scoring function is related to scoring functions employed in the link-prediction methods that the cocrystal network is used for.[3] The main difference being that this score is not designed to predict compatibility between i and j, but rather how well the edge (i, j) fits into the entire bipartite network.

Three-cycles are identified by going through E, and checking for each edge (i, j) whether there exists a vertex k such that $A_{i,k} = A_{j,k} = 1$. If such a k exists, then the vertices i, j, k form a three-cycle. To remove the cycle, we give the edges (i, j), (j, k), and (i, k) each a score using eqn (2) and remove the edge with the lowest score. An example of this is shown in Fig. 2. If two edges have the same score, one is removed at random.

The coloured three-cycles in Fig. 2 are not the only ones in the example network, but removing the two red edges is enough to eliminate all other three-cycles as well. Because the three-cycles are identified in essentially random order, it is also possible that more than one edge ends up being removed from a three-cycle. This can happen if one of the two edges that are not removed as part of one three-cycle is later removed as part of an adjacent three-cycle. This excess removal is dealt with in step 3.

In general, removing all three-cycles is not enough to fully bipartize a network. The obvious next step would be to remove five-cycles and higher order odd cycles until the network is bipartite. However, as stated previously, identifying cycles longer than three takes exponentially more time. Even finding five-cycles takes on the order of 10^6 times as many calculations, so this is not an attractive option.

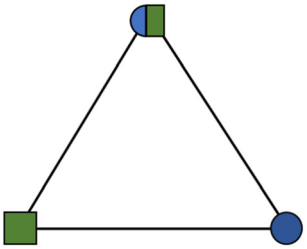

Fig. 1 Three-cycles cannot be bipartite because once any two vertices are assigned to a set (green squares or blue circles); the third vertex cannot be assigned to either set without breaking the bipartisation.

Instead, step 2 is to continue removing the lowest scoring edge or edges in the network until full bipartisation is achieved. This is done by calculating the score for every remaining edge in the network using eqn (2), removing the edge(s) with the lowest score, and then recalculating the score for the remaining edges. Because all three-cycles have been removed in the first step, the second term in eqn (2) can be left out in step 2 to save time. Finally, the network is tested for bipartiteness using the "is_bipartite()" function in the Networkx python package.[14] This cycle repeats until we are left with a perfectly bipartite subnetwork which we will call the bipartite 'seed' (Fig. 3).

The removal process in steps 1 and 2 does not immediately lead to the desired bipartite network. In our tests on the CSD cocrystal network over 70% of the edges are removed in steps 1 and 2, but it does leave us with a seed network comprised of the most well-behaved edges.

Step 3 is to start with this seed and regrow a bipartite version of the original network. This is done by calculating the score in eqn (2) for all edges in the original network (before bipartisation), and then attempting to restore each removed edge in order of score from highest to lowest. In each attempt, the edge is restored and the network is tested for bipartiteness. If the network is still bipartite, the edge is kept, otherwise, it is removed permanently.

The order in which we restore edges matters, as the bipartisation is often only broken by a combination of

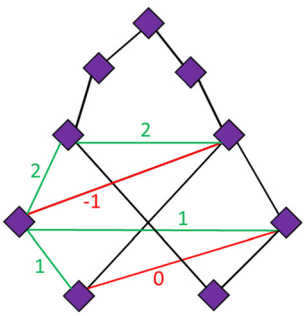

Fig. 2 An example network with two of its three-cycles marked to be removed. Each edge in these three-cycles is marked with its score. The lowest-scoring edges are marked in red.

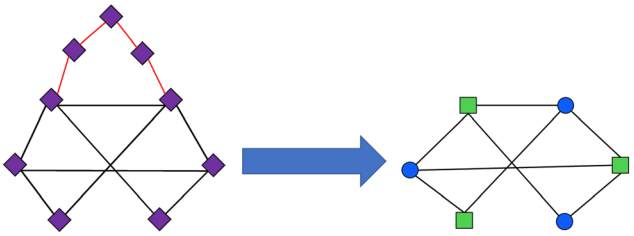

Fig. 3 An example network with the lowest scoring edges marked in red. Once these are removed, a bipartite "seed" network remains.

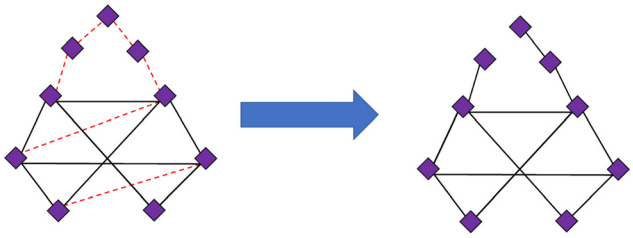

Fig. 4 An example network with its removed edges marked with red dotted lines. Edges are restored until no more edges can be restored without breaking the bipartite nature of the network.

multiple edges. Consider the four edges at the top of Fig. 4, the bipartisation is only broken once all four are restored, so whichever edge comes last will not be restored. Ordering the edges by score is a sensible choice, as keeping edges that are well connected in a bipartite manner should lead to a larger network in the end. It is still possible for two edges to have the same score, in which case they are restored in the order in which they appear in E. In the example shown in Fig. 4, the four edges at the top of the network all have the same score, and will be restored in effectively arbitrary order. This means in principle any three of the four can be restored and there is no way of knowing which three are the 'correct' ones. In a larger network a scenario like this is less likely as complex structures should cause the scores to vary more.

To visualise the bipartisation, the now bipartised adjacency matrix A can be reordered by permutation of V such that all vertices that belong to the same bipartite set are grouped together. In principle, this final reordering is not required for the bipartisation of the network, but it makes visual examination of the adjacency matrix easier by clearly splitting the adjacency matrix into the two bipartite sets. The reordering is done by placing all vertices in V_1 first and all vertices in V_2 last. To get a clearer view of the structure of the network, the vertices in V_1 are arranged in increasing order of their number of bipartite edges. The vertices in V_2 are arranged in decreasing order of their number of bipartite edges. This way the vertices with the largest number of bipartite edges are grouped together in the 'center' of the matrix. The same reordering can be applied to the original matrix O. This will result in two block matrices of the following shapes

$$O = \begin{pmatrix} M_1 & B_1 \\ B_2 & M_2 \end{pmatrix} \quad A = \begin{pmatrix} 0 & B_1 \\ B_2 & 0 \end{pmatrix}, \tag{3}$$

where M_i refers to the monopartite edges between vertices in the i-th bipartite set, and B_i refers to the bipartite edges starting from the i-th bipartite set. In undirected networks like the one we use, B_1 and B_2 are mirror images of one another. Note that B_1 and B_2 appear in both O and A. This is because the only difference between the bipartised matrix A and the reordered original matrix O is that the edges that do not fit in the bipartisation (the monopartite edges in M_1 and M_2) have been removed, so the other parts of the matrices are identical.

Appendix A contains a pseudocode description of the bipartisation algorithm.

2.4 Clustering

To further examine the structure of the cocrystal network, we can use a clustering algorithm to split the two bipartite sets into clusters of vertices that behave in similar manners. This clustering is done by first calculating the one-mode projections of the two bipartite sets.[15] Modules for calculating one-mode projections are available in the Networkx package for python. Calculating the one-mode projections will result in two graphs, each containing all vertices from one of the two bipartite sets. In these one mode projections we use eqn (4) to calculate the weights of the edges:

$$w(i,j) = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}. \tag{4}$$

Here $N(i)$ refers to the neighbours of vertex i: the set of vertices that share an edge with vertex i in the original network (before bipartisation); eqn (4) is also referred to as the Jaccard index. These weights are then used to calculate a distance matrix, where the distance between two vertices is simply $d(i, j) = 1 - w(i, j)$. Next we use Ward's hierarchical clustering method:[16] each vertex starts in its own cluster, then the two clusters that have the lowest distance are combined into one cluster. Next, the distance between the new cluster and all the older clusters is calculated according to

$$d(p,q) = \sqrt{\frac{|q|+|s|}{|q|+|s|+|t|} d(q,s)^2 + \frac{|q|+|t|}{|q|+|s|+|t|} d(q,t)^2 - \frac{|q|}{|q|+|s|+|t|} d(s,t)^2}, \tag{5}$$

where p is the new cluster created by combining clusters s and t, and q is one of the old clusters. This cycle of combining the closest clusters and recalculating the distances is repeated until exactly two clusters remain. This is done for the one-mode projections of both bipartite sets and we are left with four clusters in total. We choose to stop at two clusters because we found in our tests that adding more clusters did not lead to a meaningful partitioning. It is possible that more insight can be gained from stopping at different numbers of clusters when examining other networks.

3 Results and discussion

Before we discuss the CSD cocrystal network described in section 2.2, we want to check that our algorithm can also bipartise networks of similar size within a reasonable amount of time, even when they are not as close to being bipartite as the CSD cocrystal network. For our first test we take the bipartised cocrystal network and use its now known bipartite groups to add new edges that go against the current bipartisation. We add one random 'monopartite' edge for every two bipartite edges in the CSD network. This results in a new network with the same number of vertices and nearly 1.5 times the number of edges. The algorithm was able to bipartise this new network in approximately twice the time it took for the original network. As we will show later, the CSD cocrystal network is approximately 96% bipartite, meaning we need to remove 4% of the edges to bipartise it. If the bipartite sets we found in the CSD network were used for this new 'monopartised' network, we would find that it is approximately 64% bipartite because the number of bipartite edges has not changed, but the total has increased by a factor of 1.5. In reality, the algorithm finds an alternative division and can bipartise the network while retaining approximately 80% of the edges.

As a second test we generate a random fully connected network with the same number of vertices as the CSD cocrystal network, but four times the number of edges. It took about five times as long to bipartise this network, but it was still possible to obtain a bipartisation that retains approximately 60% of the edges despite the complete lack of inherent bipartite structure. The theoretical minimum is 50% for any network, even networks where every vertex connects to every other vertex, so 60% is not bad for a random network.

Now we move on to the CSD cocrystal network. Visualising a graph with over 8000 vertices in such a way that information can still be gathered is not realistic. Instead we choose to visualise the adjacency matrix in a scatterplot where each dot represents a 1, and 0's are represented by empty spaces. Such a scatterplot for the original cocrystal network is shown on the left hand side of Fig. 5.

The matrix is symmetrical, which it should be as our network is undirected. This adjacency matrix shows no sign of bipartiteness yet. One notable feature is the apparent line along $x = y$. This line, however, is an illusion caused by the finite dot size and the fact that the matrix is symmetrical in the line $x = y$. If there is a dot very close to this line, then there must be another dot on the other side. These dots can overlap, giving the appearance of a dot on the line itself. Dots close to the line are relatively common because of the way the network is constructed; if a cocrystal is added and neither of its coformers are already in the network, those coformers will appear side by side in the adjacency matrix. One might also notice what looks like a number of slightly denser clusters near the bottom-right of the figure, but these are again caused by the fact that similar compounds often have similar refcodes in the CSD. Because cocrystals are added in alphabetical order of refcodes, these similar compounds are often placed close to one another. We can get rid of these artifacts by randomising the order of the coformers as is shown on the right hand side of Fig. 5.

The original network contains 8506 vertices connected by 11 668 edges. After steps 1 and 2 of the algorithm have finished, we are left with a seed network of 823 vertices connected by 3252 edges. This seems like a poor result, as most of the network was lost, but by applying step 3 we can recover almost all of the removed edges. After step 3 has completed, we once again have 8506 nodes now connected by 11 200 edges. Thus,

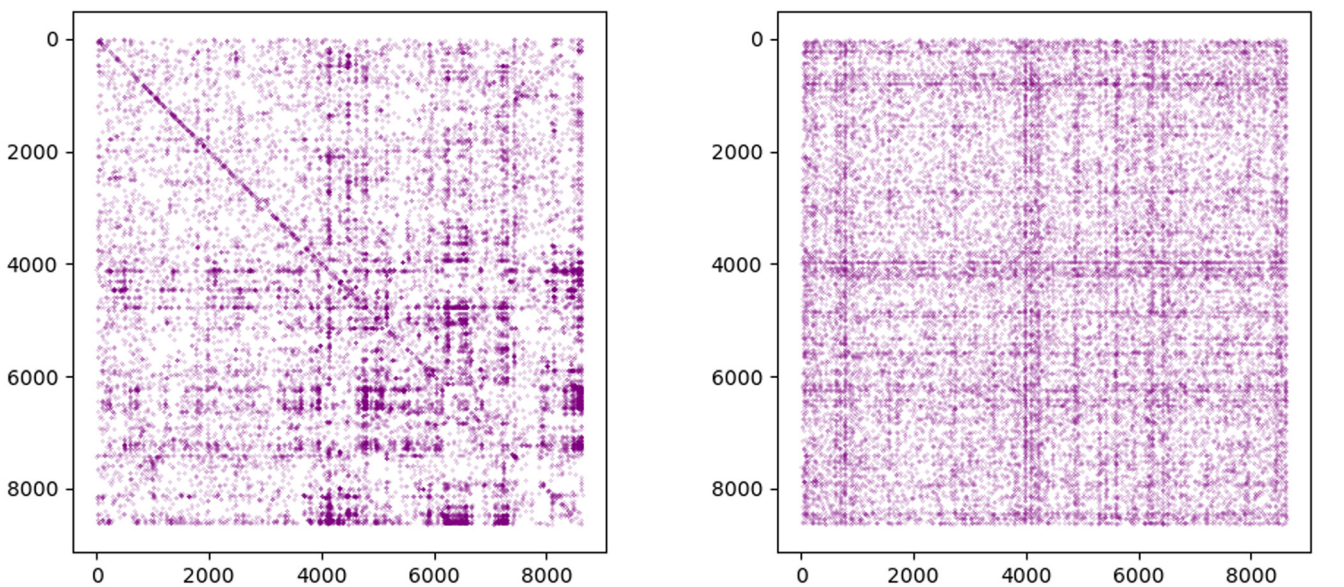

Fig. 5 (Left) A scatterplot of the original adjacency matrix belonging to the CSD cocrystal network. Each dot represents the position of a 1 (cocrystal) in the adjacency matrix. Note that the y-axis increases downwards as it would in a matrix representation. (Right) A scatterplot of the same network after the order of labels of the coformers has been randomised.

nearly 96% of the original edges are still in the network. Since our algorithm is not exact, it is possible that there is a way to retain even more of the network after bipartisation. Therefore, 96% is a lower bound for the percentage of bipartite edges. Note that the estimate made by Devogelaer *et al.*[5] that 99% of edges behave bipartite was based on the number of bipartite and monopartite paths around an edge. This is a somewhat heuristic way of estimating how close the network is to being bipartite, and is based only on local analysis. What we can measure here is a hard lower bound, rather than a rough estimate. Because so many edges are left after bipartisation, we can now say that the bipartite behaviour found by Devogelaer *et al.* is not just a local property of individual links, but a global property of the entire network.

After the bipartisation, we reorder the vertices such that the bipartisation becomes more apparent in the adjacency matrix. The result of the bipartisation is shown in Fig. 6.

Fig. 6 shows a clear block matrix structure with two empty blocks and two mirrored blocks filled with edges. The size of the two bipartite sets can be seen by looking at the dimensions of the filled blocks. Since the network has been bipartised, any edge in the network has one end in V_1 and one end in V_2. In the same way any dot in Fig. 6 must have one coordinate in V_1 and one in V_2. Within each bipartite set, the vertices are ordered such that those with the most edges to the other set are placed closest to the middle as described in section 2. This leads to the cross shape visible in Fig. 6. The bipartite sets are approximately equal in size at just over 4000 vertices, suggesting most of the coformers behave similarly with respect to the bipartite nature of the network. To show why that is more surprising than one might think, we examine the degree distribution of the network. Looking at the degree distribution of the network in Fig. 7, it is clear that there is a small number of vertices that are very well

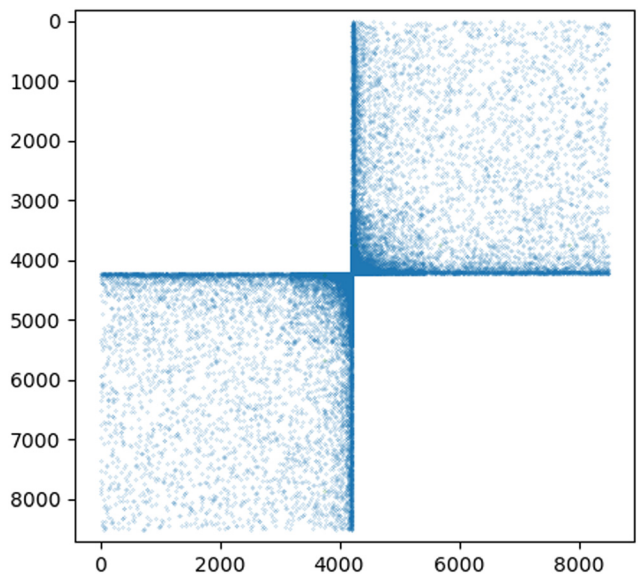

Fig. 6 The adjacency matrix of the bipartised cocrystal network. The vertex labels are marked on both *x*- and *y*-axes.

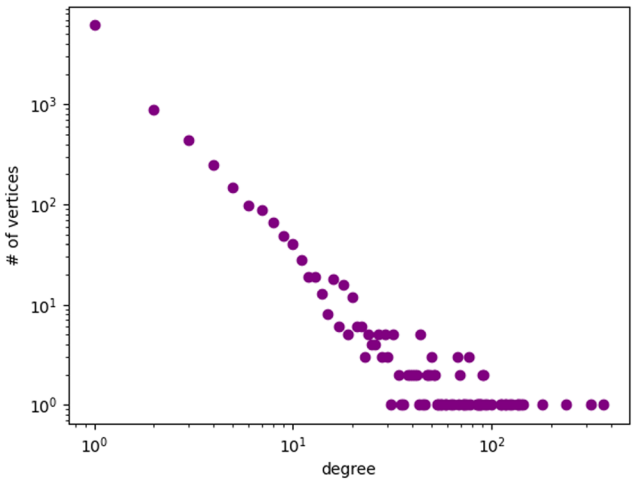

Fig. 7 Degree distribution of the CSD cocrystal network in log–log scale.

connected (the bottom right of Fig. 7), while the vast majority of vertices have at most a handful of edges (the top left of Fig. 7).

One might expect the really well-connected vertices to form one bipartite set, as they must each connect to a very large number of poorly-connected vertices, which would then form the other bipartite set. In that case, one bipartite set would have been far larger than the other, and we would conclude that the coformers represented by the well-connected vertices share some unique quality that sets them apart from the rest. However, Fig. 6 clearly shows that the bipartite sets are nearly equal in size. This instead tells us that as a group, the well-connected vertices, and the coformers they represent, behave no differently than all the others. We can confirm this by noting that the degree distributions of the individual bipartite sets shown in Fig. 8 have the same shape as the degree distribution of the entire network in Fig. 7.

Our bipartisation algorithm leaves all vertices intact. This means we can permute the original matrix, including all the removed edges, to match the way we reordered the bipartised matrix. This will allow us to clearly visualise the near-bipartite nature of the original CSD cocrystal network. The result of this reordering is shown in Fig. 9.

To make the bipartite behaviour more obvious, edges are marked in blue if they connect the two bipartite sets defined by the bipartised network (96%), and in red if they lie within only one of the two bipartite sets (4%). It is plain to see that the network was indeed close to bipartiteness to begin with. The two bipartite blocks are much more densely populated than their 'monopartite' counterparts. An interesting feature is that the densest population of monopartite edges coincides with the 'center' of the matrix, where the vertices with the most bipartite edges are concentrated. This suggests that a lot of the non-bipartite edges come from coformers that have been used quite often in successful experiments. It is tempting to think that these coformers show 'universal' behaviour because they are used more often in experiments. However, many of the coformers near the center, which are the coformers that are used the most, have no monopartite edges at all. For example,

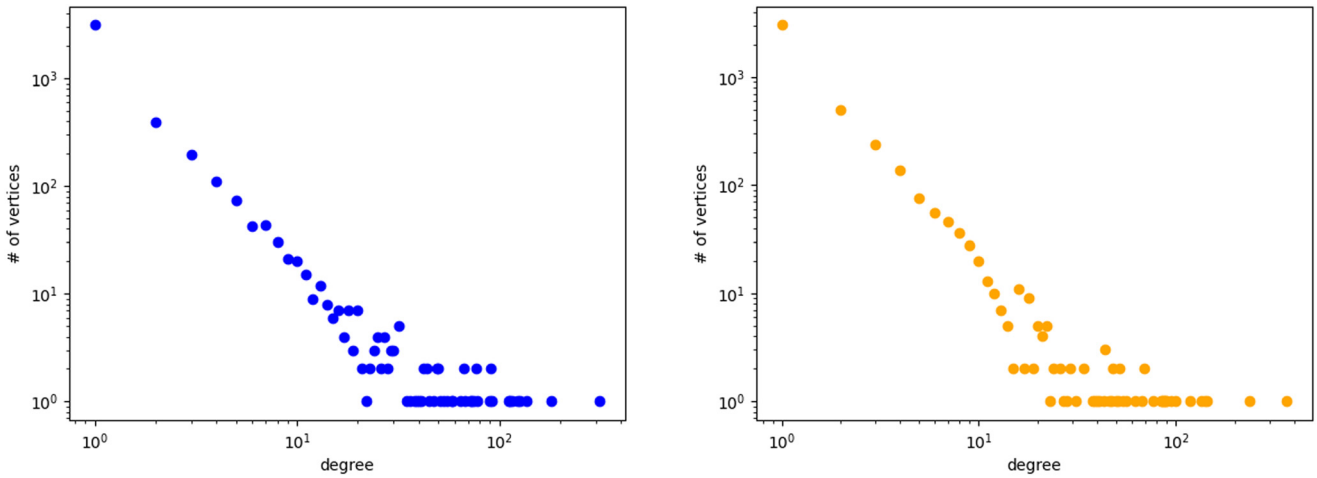

Fig. 8 The degree distributions of the two bipartite sets. Their shape is the same as the one for the entire network. This means the bipartite structure of the network is independent of the degrees of the vertices.

tetramethylpyrazine has 51 edges, none of which are monopartite. This should mean that there are some coformers that have some kind of universal quality that make them capable of cocrystallising with a large variety of other coformers, even if they lie in the same bipartite set. However, the bipartite nature of the network as a whole shows that overall, the coformers cocrystallise in complimentary pairs.

We can now identify the coformers that have the highest number of 'monopartite' edges. These 'universal' coformers do not behave in the same way as the others within the network. The term 'universal' suggests that these coformers are not selective at all, however, most of these universal coformers still have a lot more bipartite than monopartite edges. It is expected that they have some chemical characteristics that explain this universal behaviour. To examine this we extract the nine most monopartite coformers from both bipartite sets. These coformers are shown in Fig. 10 and 11. Most of these 'universal' coformers do indeed have groups that can serve as hydrogen bond donors, as well as groups that can serve as hydrogen bond acceptors. It is likely that a large number of coformers with only hydrogen donor groups belong to one bipartite set, and a large number with only hydrogen acceptor groups belong to the other. This would account for some of the bipartite nature of the network, and why these universal coformers have so many monopartite edges.

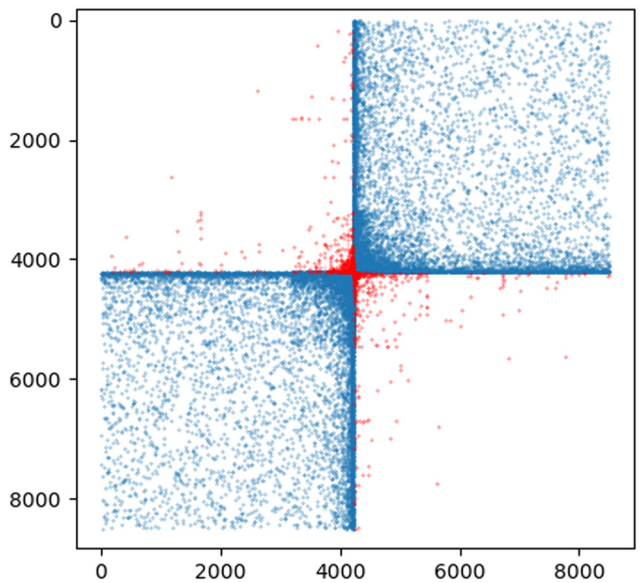

Fig. 9 The adjacency matrix of the full non-bipartised network, reordered in the same way as the bipartised network. Bipartite edges are marked in blue, monopartite edges are marked in red.

There are a few coformers, like benzoquinone and triphenylphosphine oxide, that do not have both hydrogen bond acceptor and donor groups. These coformers must have another universal property that allows them to go against the bipartite nature of the network. Benzoquinone for example, has two hydrogen bond acceptor groups, but no donor groups. We examined all benzoquinone cocrystals found in the network, specifically looking for coformers that cannot form hydrogen bonds with benzoquinone. All of these coformers that we extracted from the CSD have the ability to form π–π interactions. It seems that these π–π interactions are how benzoquinone gets around the bipartite nature of the network.

Identifying the universal coformers can be very useful in cocrystal screening. On one hand, they can be removed from the network to help improve the accuracy of computational prediction techniques. On the other hand, the fact that they cocrystallise with either bipartite set means they are good first choices for experimental cocrystal screening. Lists of the most popular coformers in each bipartite set (those with deg ≥ 20) are available as ESI.† These lists are ordered by decreasing number of monopartite edges, and then by increasing number of bipartite edges. This way the most universal coformers are at the top and the least universal ones are at the bottom.

We can now use a clustering algorithm to divide the two bipartite sets further in an attempt to discover further structure in the network. The clustering algorithm we use requires a fully

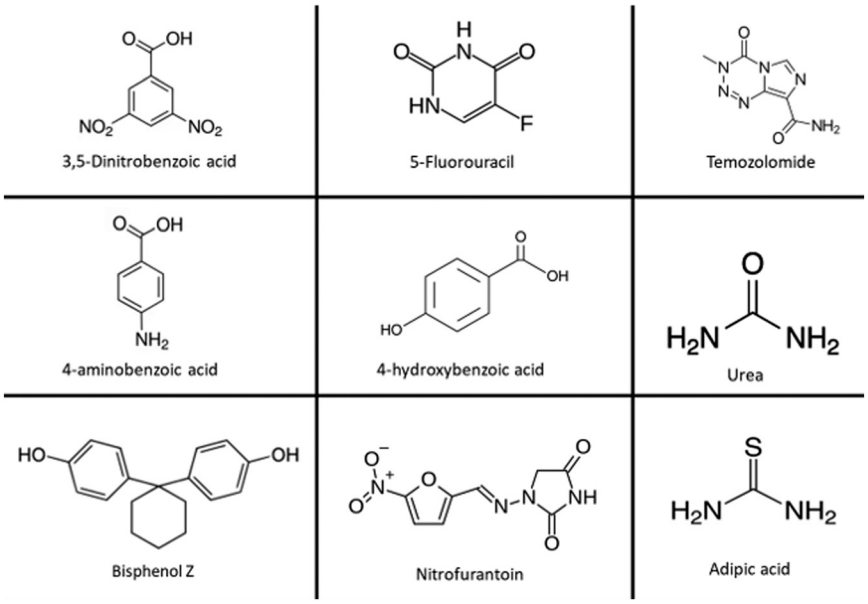

Fig. 10 The nine 'most universal' coformers with the highest number of monopartite edges in the first bipartite set.

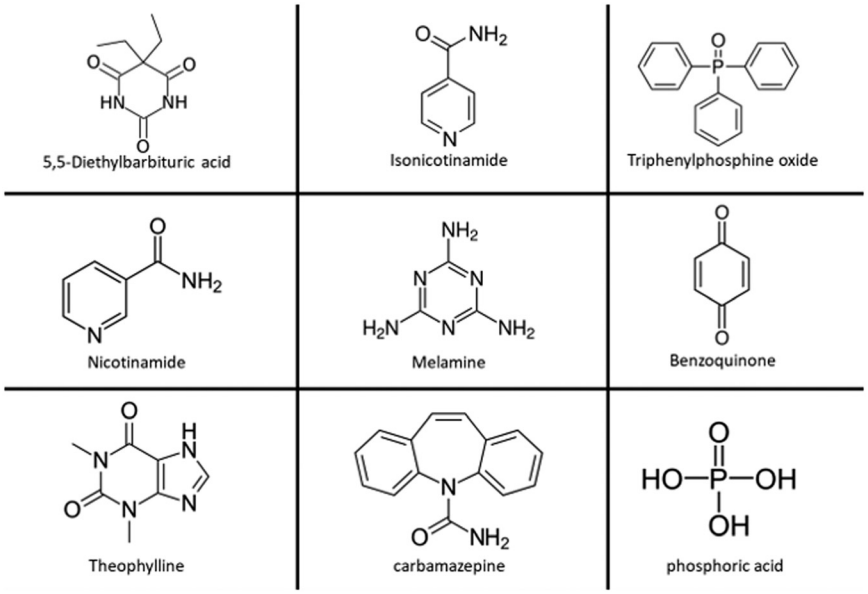

Fig. 11 The nine 'most universal' coformers with the highest number of monopartite edges in the second bipartite set.

connected network. A network is considered fully connected if every pair of vertices has a path of finite length from one to the other. Since our network is not fully connected we extract the largest connected subnetwork and bipartise it separately. Examining the adjacency matrix for this largest subnetwork, shown in Fig. 12, an interesting feature stands out. Both of the blue bipartite blocks have a large rectangular void. This seems to suggest that the bipartite sets A and B could be split into two smaller sets A_1, A_2, B_1, and B_2, such that vertices from A_1 and B_1 never connect to one another. However, this is another illusion. Because the matrix is ordered by the number of bipartite edges, all the vertices with only one bipartite edge are grouped together in both bipartite sets. If two vertices from opposite bipartite sets both have only one bipartite edge, then they form their own separate subnetwork, disconnected from the rest of the network. Because we are only looking at the largest connected subnetwork, such vertices will have been removed, forcing a void like this to exist. What this does show is that the network contains a large number of what we will call 'dendrites': vertices attached to the larger network by only a single edge. These dendrites do not contribute to the bipartisation of the network as you can always attach a dendrite to a bipartite network without breaking bipartiteness. On the other hand, their lack of neighbours can cause issues when using Ward's clustering algorithm, because

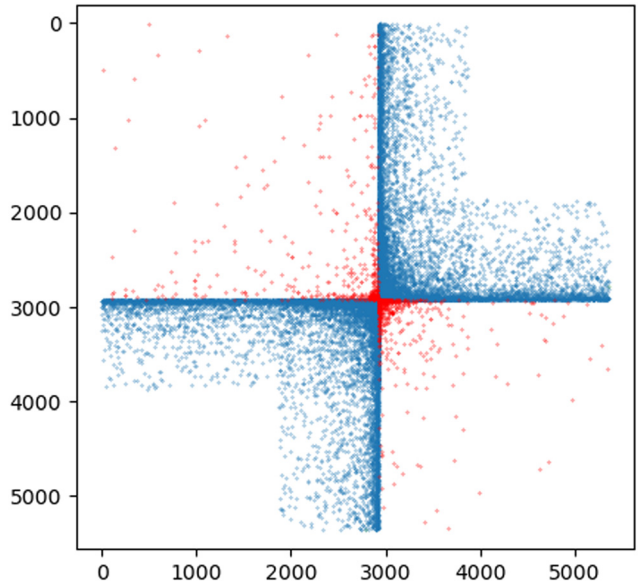

Fig. 12 The adjacency matrix of the largest connected subnetwork of the cocrystal network.

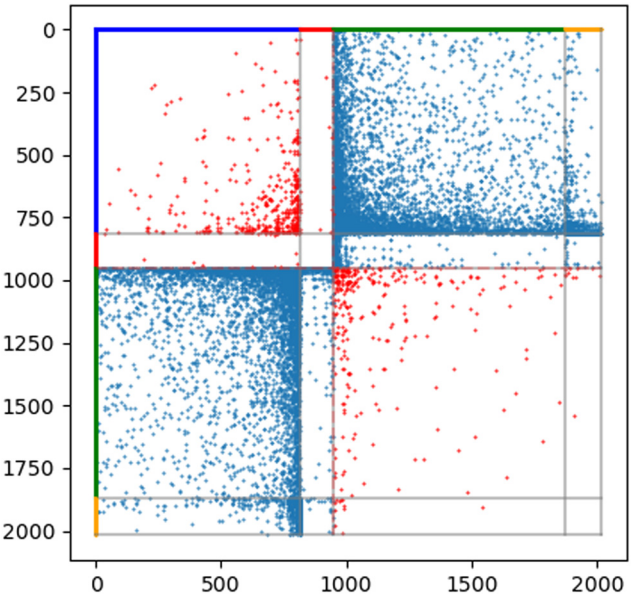

Fig. 13 The adjacency matrix of the clustered network, each cluster is represented by a different colour on the axes.

there is not enough information to properly place them in a cluster. To help the clustering algorithm work properly we decided to apply it only to vertices with a degree larger than 1.

Now that we have the largest connected subnetwork with degrees larger than 1 we can use the one-mode projections and Ward's clustering algorithm to split the two bipartite sets into two smaller clusters. Once the clustering algorithm is finished we again reorder the adjacency such that the newly created clusters are separated from one another. Fig. 13 shows the clustered adjacency matrix, the colours along the x- and y-axes indicate the four clusters.

The clustering shows that both bipartite groups have one cluster that does not have a single monopartite link to itself (the red and yellow clusters). This means that the clustering algorithm has managed to identify a subsection of the network that was already perfectly bipartite even before bipartisation. Both the red and yellow clusters are examples of so called anticommunities: clusters of vertices that do not connect to one another but do connect to vertices outside the cluster. The ability to find anticommunities is desirable in certain fields of research,[12] but we can also imagine practical applications in cocrystal screening. Because none of the coformers in the red cluster cocrystallise together, it is possible to make a 'cocktail' of all of them combined in one solution. Adding a target coformer to this cocktail solution would screen it for cocrystals with any of the coformers in the red cluster in a single experiment.

It should be noted that in principle, experimental verification of these anticommunities is needed to prove that absolutely no cocrystals can be formed using the coformers in these anticommunity sets. The CSD network only represents cocrystallisation experiments that have already been performed and reported in literature.

A full list of coformers found in the red and yellow clusters, the anticommunities, is provided as ESI.†

4 Conclusions

We have successfully designed an algorithm that can approximate an optimal bipartisation for large networks. Because of the inexact nature of the algorithm, it is able to bipartise much larger networks in a reasonable amount of time, unlike algorithms designed to exactly find the optimal bipartisation. By bipartising the network of cocrystals taken from the Cambridge Structural Database(CSD), we were able to more closely examine its structure. We showed that this network is at least 96% bipartite, and that its bipartite behaviour is a global property of the network. We were able to show that the bipartite behaviour is independent of the degree of vertices, and we can identify coformers that behave 'universally' rather than conform to the bipartite structure. We found that the network has a relatively dense 'core' of really well-connected vertices and a very large number of 'dendrites' branching off from that core. Finally we were able to use one-mode projections and Ward's clustering algorithm to split the bipartite sets into smaller clusters. This allowed us to identify two anticommunities that form a perfectly bipartite subnetwork together from the start. Since the coformers in each of these groups do not interact with other coformers in the group, they could be used in a 'cocktail method' for cocrystal screening. These anticommunities could not have been identified without bipartisation of the entire network.

Conflicts of interest

There are no conflicts to declare.

A Appendix: bipartisation algorithm

What follows is a pseudocode description of our bipartisation algorithm.

Algorithm 1 Bipartisation

Require: Adjacency matrix A, list of vertices V and list of edges E

$O \leftarrow A$ ▷ O will hold a copy of the original adjacency matrix

==========

Step 1:

==========

for all $(i, j) \in E$ **do**

  while $\exists k : \{(i, k), (j, k)\} \subset E$ **do**

    Score(i,j)$\leftarrow \sum_{(n,m)\in E} (A_{i,n}A_{j,m} + A_{i,m}A_{j,n})\text{-}\sum_{l=1}^{N} (A_{i,l}A_{j,l})$

    Score(i,k)$\leftarrow \sum_{(n,m)\in E} (A_{i,n}A_{k,m} + A_{i,m}A_{k,n})\text{-}\sum_{l=1}^{N} (A_{i,l}A_{k,l})$

    Score(j,k)$\leftarrow \sum_{(n,m)\in E} (A_{j,n}A_{k,m} + A_{j,m}A_{k,n})\text{-}\sum_{l=1}^{N} (A_{j,l}A_{k,l})$

    Remove the lowest scoring edge from E, add it to R, and update A. ▷ R is a list of all removed edges

  end while

end for

==========

Step 2:

==========

while A is not bipartite **do**

  for $e \in E$ **do**

    Score(e)$\leftarrow \sum_{(n,m)\in E} (A_{e_1,n}A_{e_2,m} + A_{e_1,m}A_{e_2,n})$

  end for

  remove the lowest scoring edge from E, add it to R and update A.

end while

==========

Step 3:

==========

for $r \in R$ **do**

  Score(r)$\leftarrow \sum_{(n,m)\in E} (O_{r_1,n}O_{r_2,m} + O_{r_1,m}O_{r_2,n})\text{-}\sum_{l=1}^{N} (O_{r_1,l}O_{r_2,l})$

end for

Sort R from highest score to lowest score

for $r \in R$ **do**

  Add r back to E

  if A is (still) bipartite **then**

    Continue

  else

    Remove r from E

  end if

end for

==========

Reordering and visualisation:

==========

Split V into two known bipartite sets V_1 and V_2

reorder the vertices in V_1 by increasing number of edges to V_2

reorder the vertices in V_2 by decreasing number of edges to V_2

Define the unitary matrix T that transforms V into the reordered $V_1 + V_2$

$A \leftarrow TAT^T$

$O \leftarrow TOT^T$

plot A and O

References

1 E. Estrada, *The Structure of Complex Networks: Theory and Applications*, Oxford University Press, 2011.
2 M. E. J. Newman, *Networks: An Introduction*, Oxford University Press, 2010.
3 J. J. Devogelaer, S. Brugman, H. Meekes, P. Tinnemans, E. Vlieg and R. de Gelder, Co-crystal design by network-based link prediction, *CrystEngComm*, 2019, **21**, 6875–6885.
4 J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Macmillan, 1976.
5 J. J. Devogelaer, H. Meekes, E. Vlieg and R. de Gelder, Co-crystals in the Cambridge Structural Database: a network approach, *Acta Crystallogr., Sect. B: Struct. Sci.*, 2019, **75**, 371–383.
6 L. Zheng, B. Zhu, Z. Wu, X. Fang, M. Hong and G. Liu, *et al.*, Strategy for efficient discovery of cocrystals via a network-based recommendation model, *Cryst. Growth Des.*, 2020, **20**(10), 6820–6830.
7 V. Choi, Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design, *Quantum Inf. Process.*, 2010, **10**(3), 343–353, DOI: **10.1007/s11128-010-0200-3**.
8 F. Hüffner, Algorithm Engineering for Optimal GraphBipartization, *J. Graph Algorithms Appl.*, 2009, **13**(2), 77–98.
9 P. Heggernes, P. van 't Hof, D. Lokshtanov and C. Paul, Obtaining a Bipartite Graph by Contracting Few Edges, *SIAM J. Discrete Math.*, 2013, **27**(4), 2143–2156.
10 M. Pilipczuk, M. Pilipczuk and M. Wrochna, Edge Bipartization faster than 2^k, *Algorithmica*, 2017, **81**, 917–966.
11 T. D. Goodrich, E. Horton and B. D. Sullivan, An Updated Experimental Evaluation of Graph Bipartization Methods, *ACM J. Exp. Algorithmics*, 2021, **26**(1), 12, DOI: **10.1145/3467968**.
12 A. Concas, S. Noschese, L. Reichel and G. Rodriguez, A spectral mehtod for bipartizing a network and detecting a large anti-community, *J. Comput. Appl. Math*, 2020, **373**, 112306.
13 I. J. Bruno, J. C. Cole, P. R. Edgington, M. Kessler, C. F. Macrae and P. McCabe, *et al.*, New software for searching the Cambridge Structural Database and visualizing crystal structures, *Acta Crystallogr., Sect. B: Struct. Sci.*, 2002, **58**(3 Part 1), 389–397, DOI: **10.1107/S0108768102003324**.
14 A. Hagberg, P. Swart and D. Schult, *Exploring network structure, dynamics, and function using NetworkX*, Los Alamos National Lab. (LANL), Los Alamos, NM (United States), 2008.
15 S. P. Borgatti and D. S. Halgin, Analyzing Affiliation Networks, *The SAGE Handbook of Social Network Analysis*, 2014.
16 J. H. Ward Jr., Hierarchical Grouping to Optimize an Objective Function, *J. Am. Stat. Assoc.*, 1963, **58**(301), 236–244.