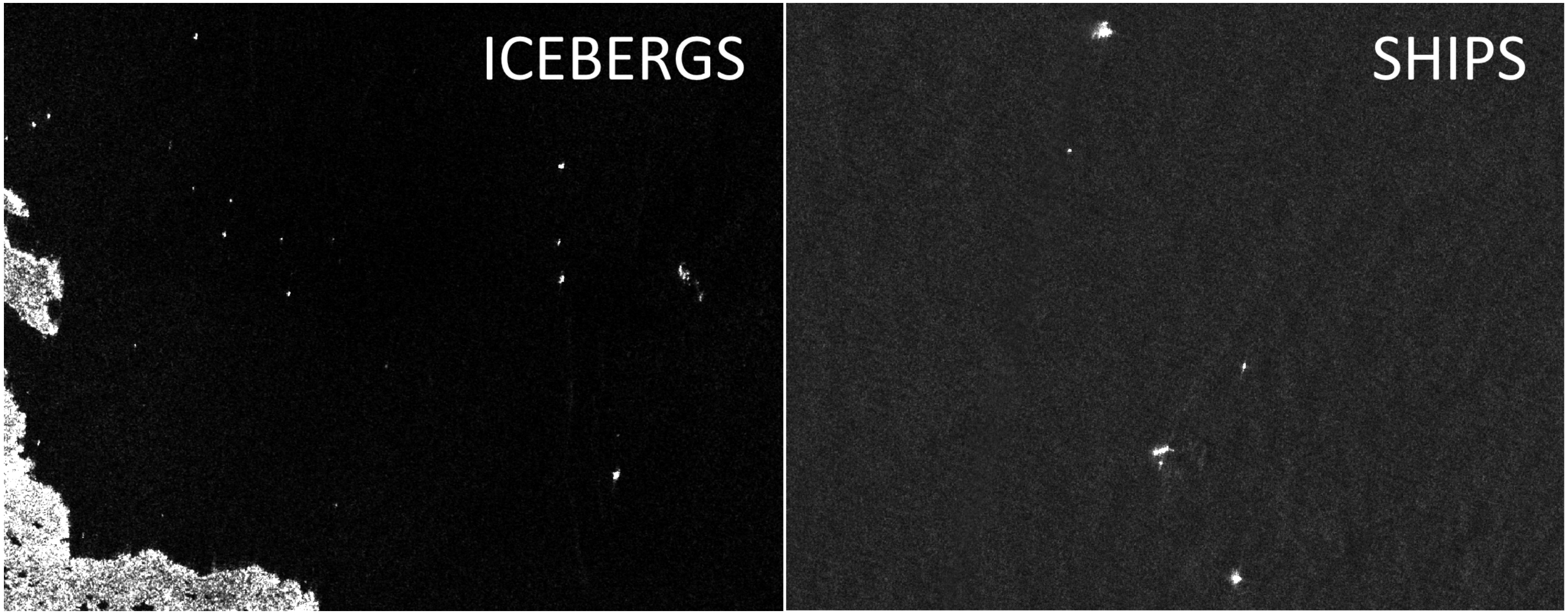
Titanic

How can deep learning on irregular domains help to save lifes?

0 - Table of content

1 - Introduction

© Statoil/C-CORE - Icebergs and ships examples



2 - Data source

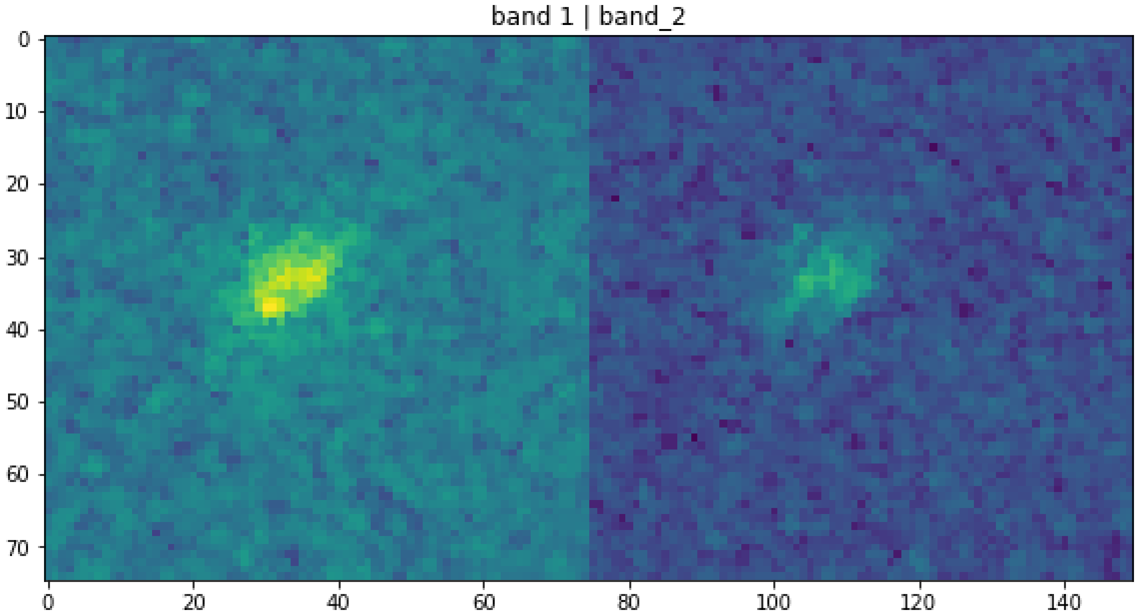
`10'028` iceberg or ship cases with only `1'604` labelled

Description

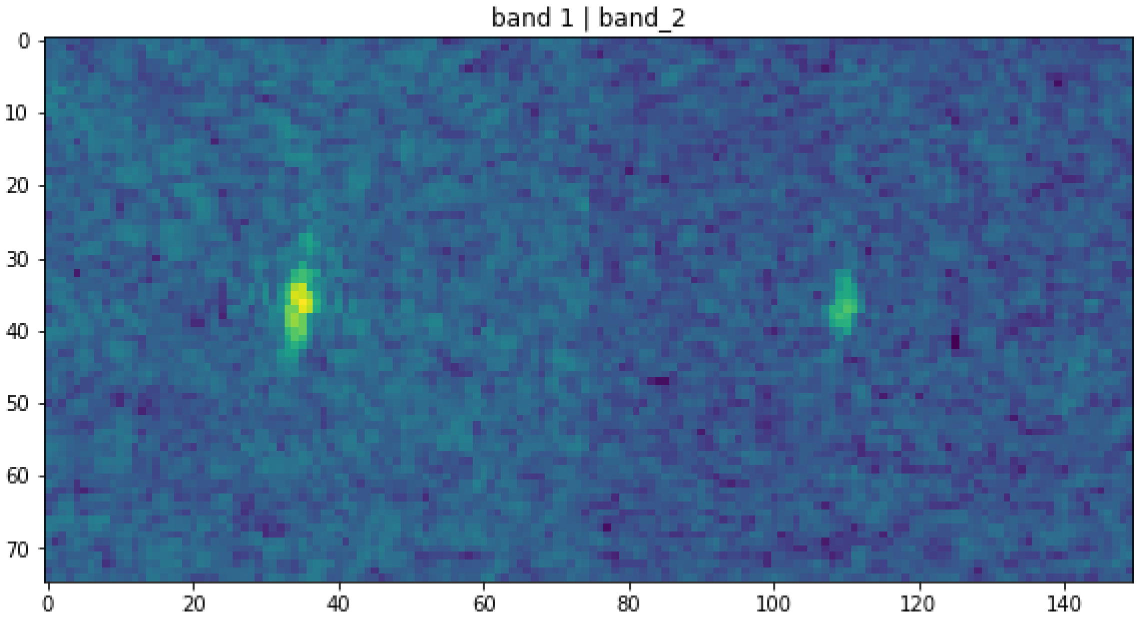
Feature	Description	Type	Has N/A	Comment
id	image identifier	String	No	
band_1	horizontal plane	Float array	No	HH
band_2	vertical plane	Float array	No	HV
inc_angle	measurement angle	Float	Yes (~10%)	Unit in degrees
is_iceberg	iceberg or not	Boolean (0/1)	No	Label

Exploration

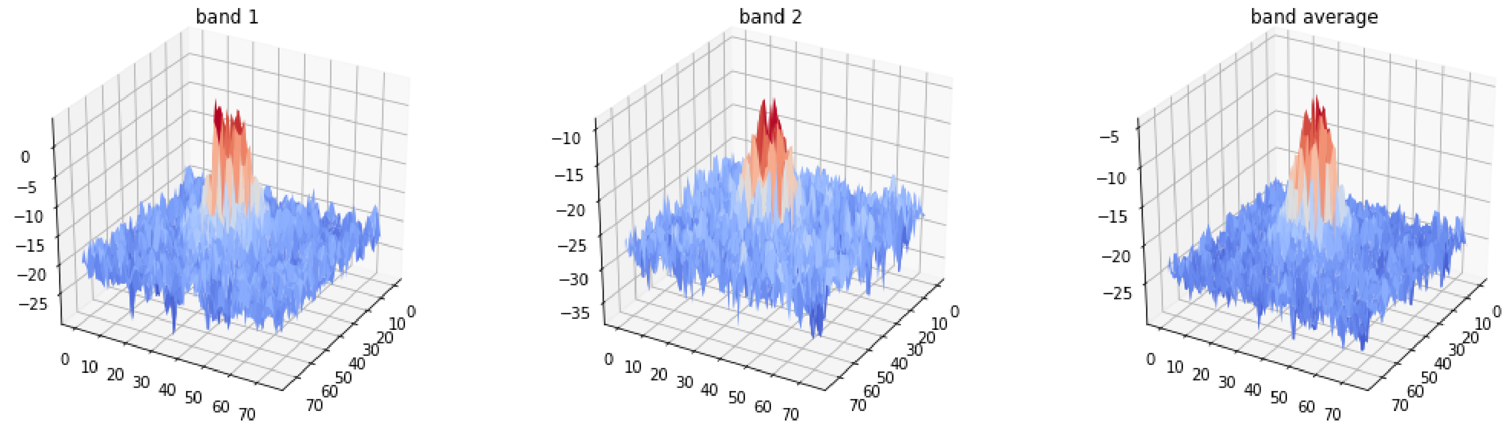
In [10]:
```
viz.plot_bands(example_iceberg)
```



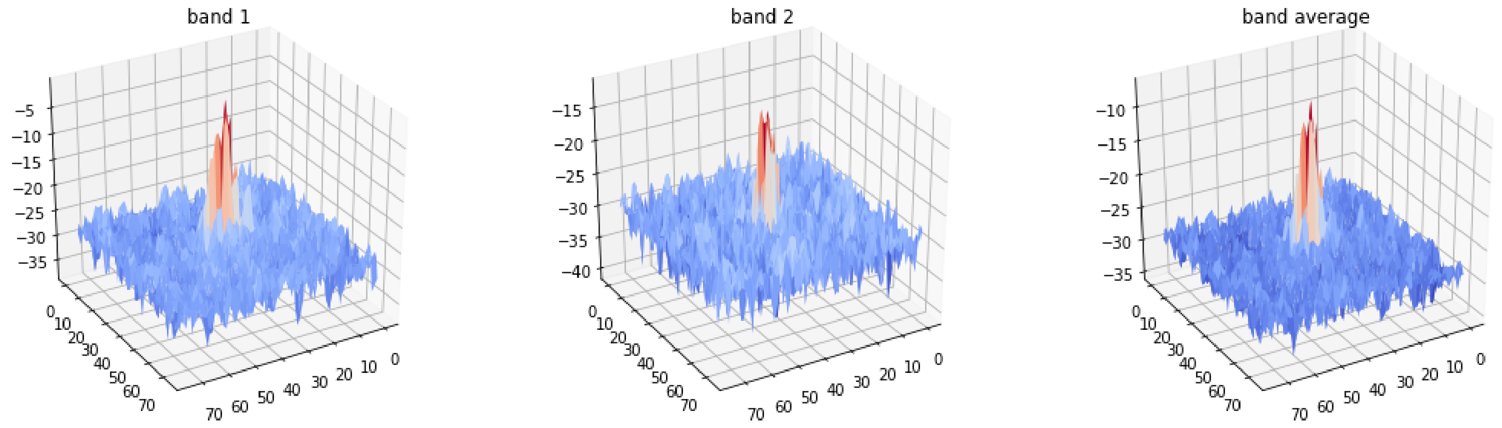
In [13]:
```
viz.plot_bands(example_ship)
```



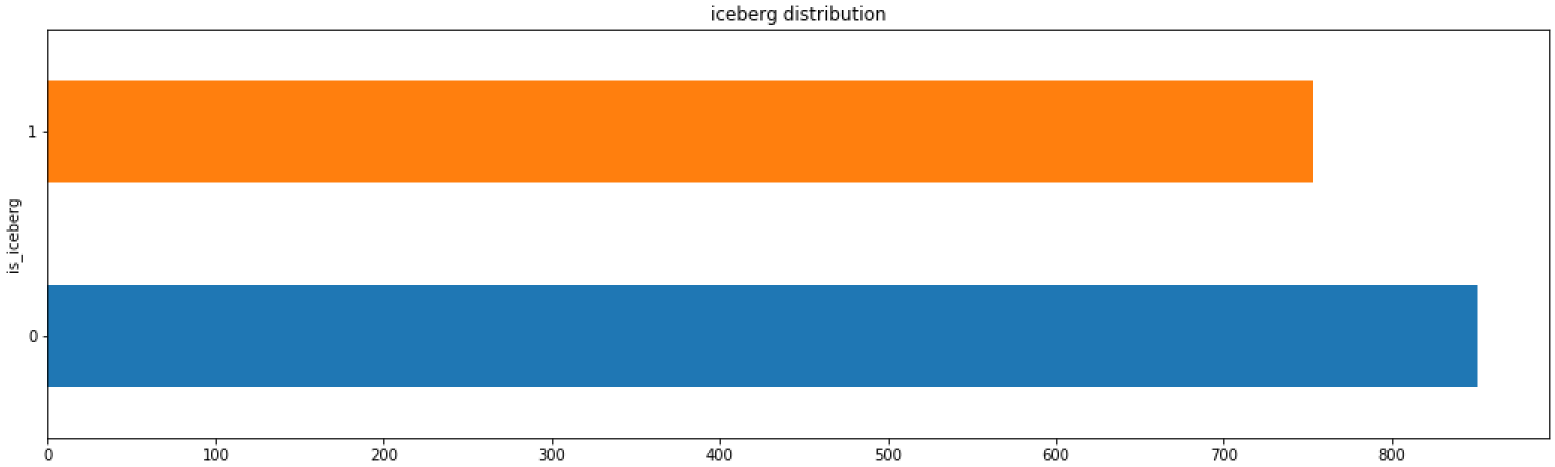
```
In [11]: viz.plot_bands_3d(example_iceberg)
```




```
In [14]: viz.plot_bands_3d(example_ship, angle=60)
```

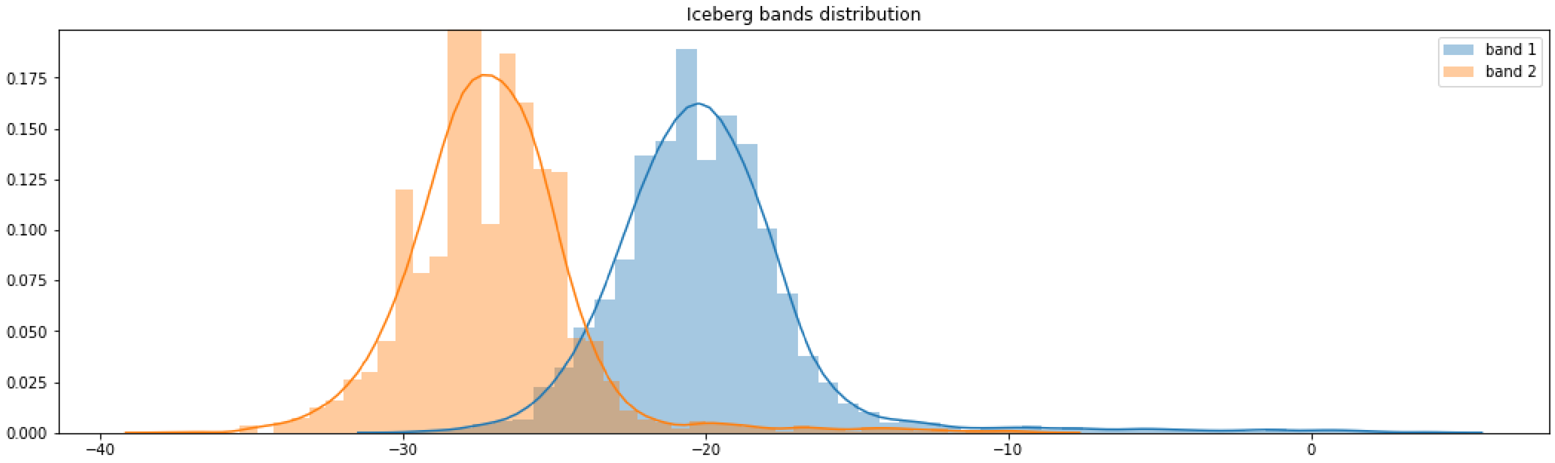


In [15]:
```
plt.title('iceberg distribution')
measures.groupby(measures.is_iceberg).is_iceberg.count().plot.barh();
```



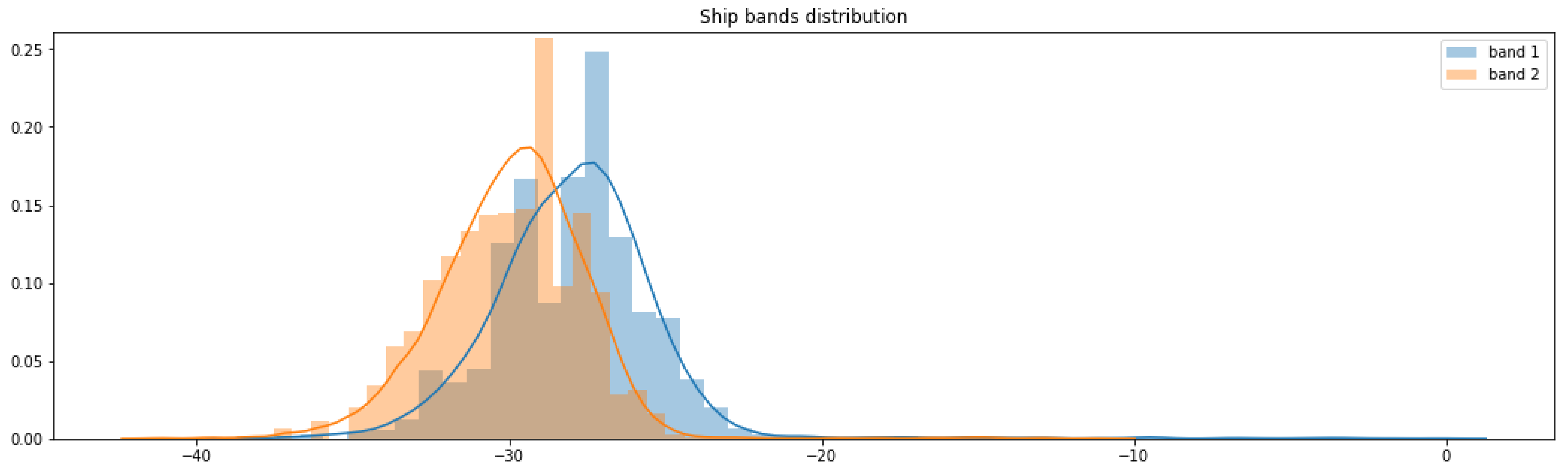
In [18]:

```
plt.title('Iceberg bands distribution')
sns.distplot(example_iceberg.band_1, label='band 1')
sns.distplot(example_iceberg.band_2, label='band 2')
plt.legend();
```

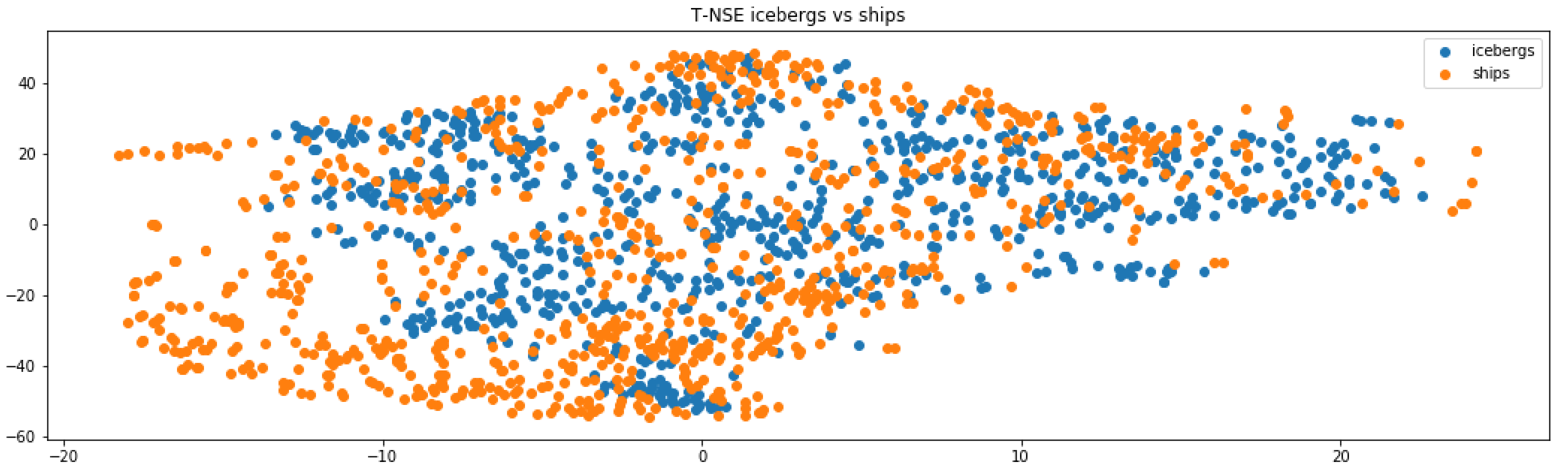


In [19]:

```
plt.title('Ship bands distribution')
sns.distplot(example_ship.band_1, label='band 1')
sns.distplot(example_ship.band_2, label='band 2')
plt.legend();
```



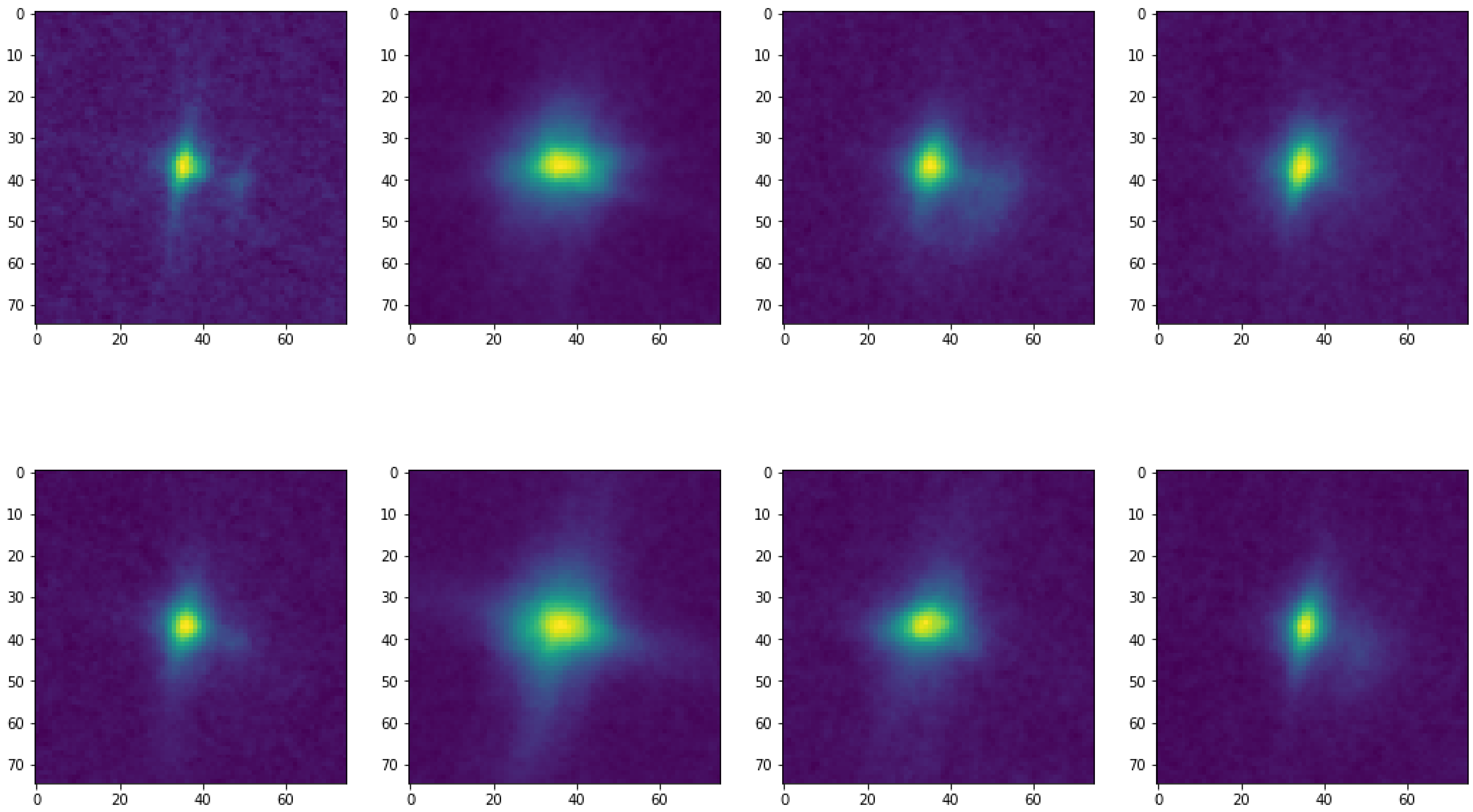
In [23]:
```
plt.title('T-NSE icebergs vs ships')
plt.scatter(tsne[measures.is_iceberg == 1, 0], tsne[measures.is_iceberg == 1, 1],
label='icebergs')
plt.scatter(tsne[measures.is_iceberg == 0, 0], tsne[measures.is_iceberg == 0, 1],
label='ships')
plt.legend();
```



Prototypes

In [26]:

```
for i, center in enumerate(kmeans_centers):
    plt.subplot(1, 4, i % 4 + 1)
    plt.imshow(center.reshape(75, 75))
    if i % 4 == 3:
        plt.show()
```



3 - Preprocessing

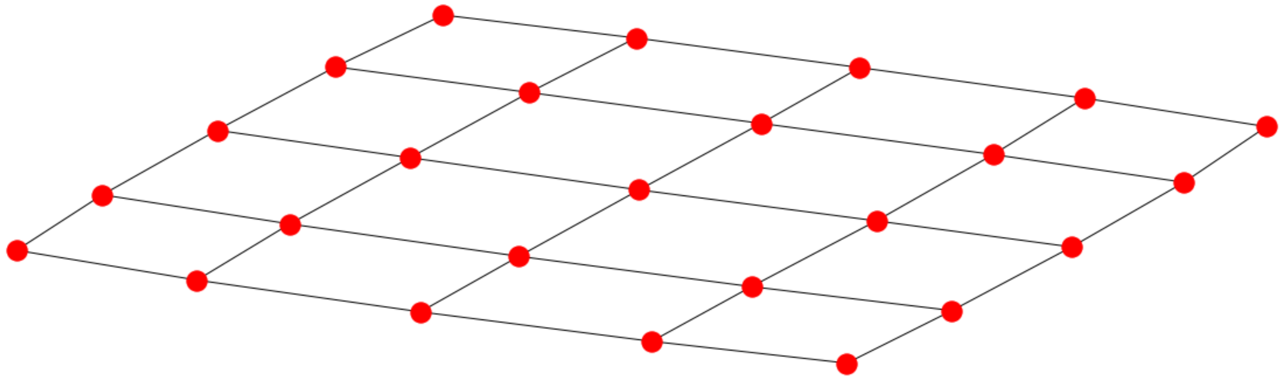
- No filtering
- No data augmentation
- Rescaling between 0 and 1
- Replacing missing angle by 0

4 - Graphs

Classical 2D grid

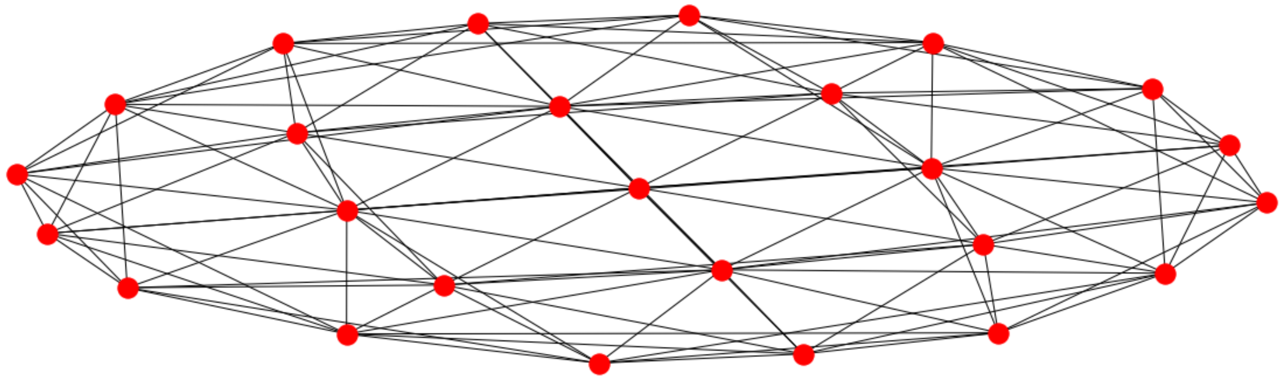
In [39]:

```
small_grid = nx.grid_graph([5, 5])
nx.draw(small_grid)
```



Knn 2D grid

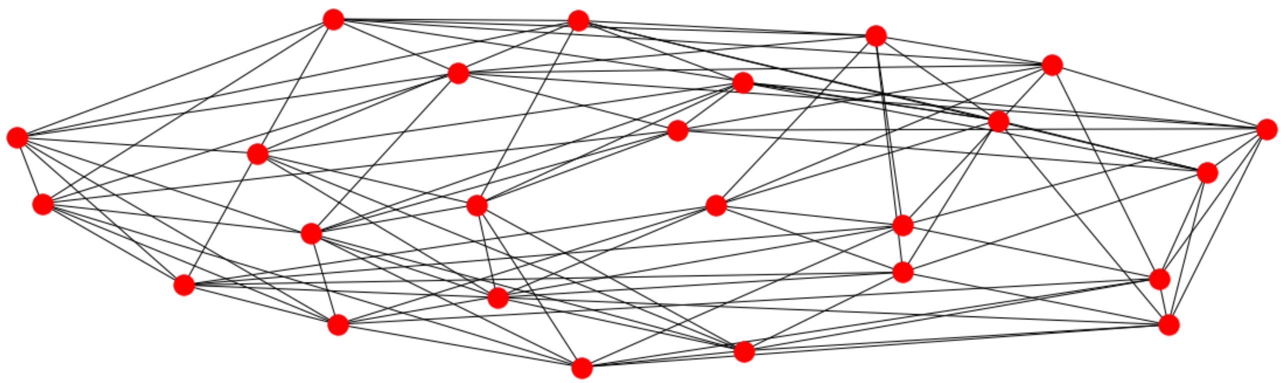
In [40]:
```
small_knn = graph.knn(graph.grid_coordinates(5), k=8, metric='cityblock')
nx.draw(small_knn)
```



Wrap-around Knn 2D grid

In [41]:

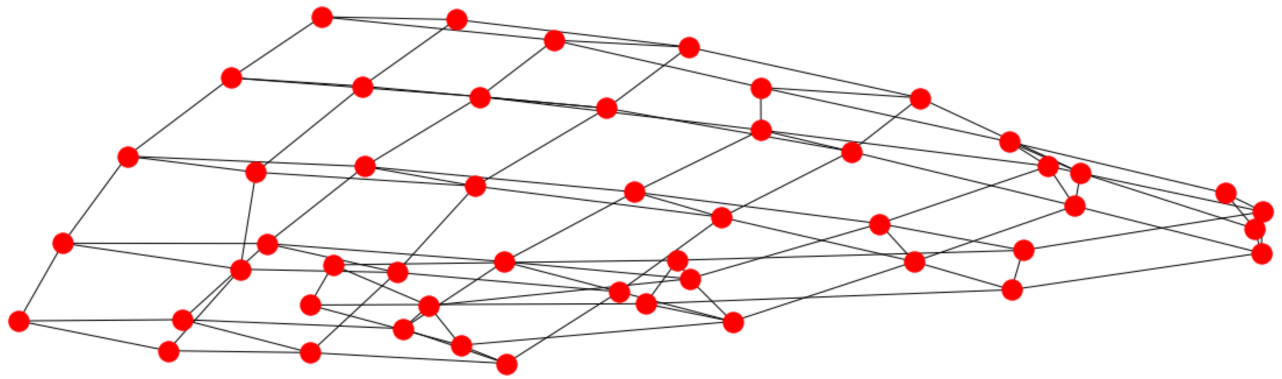
```
small_wraps = graph.kwraps(5, kd=1)
nx.draw(small_wraps)
```



Classical 3D grid

In [42]:

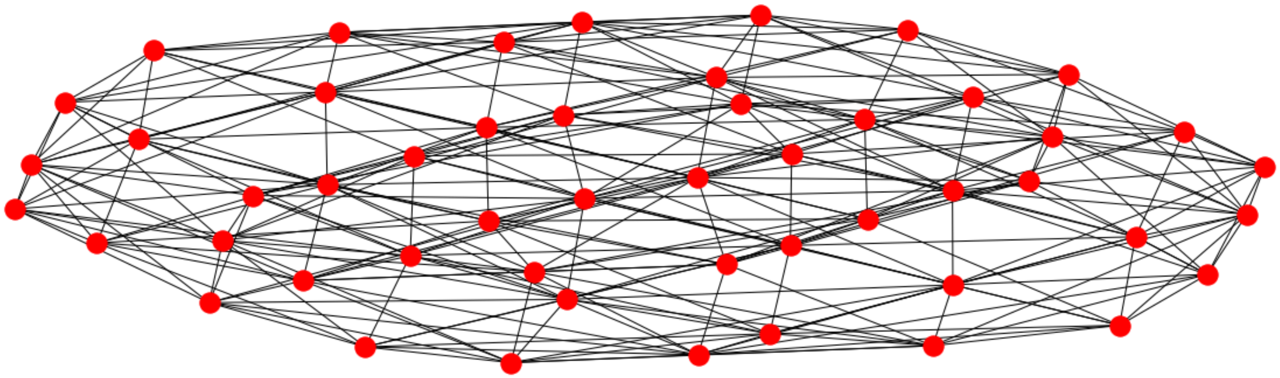
```
small_grid3d = nx.grid_graph([5, 5, 2])
nx.draw(small_grid3d)
```



Knn 3D grid

In [43]:

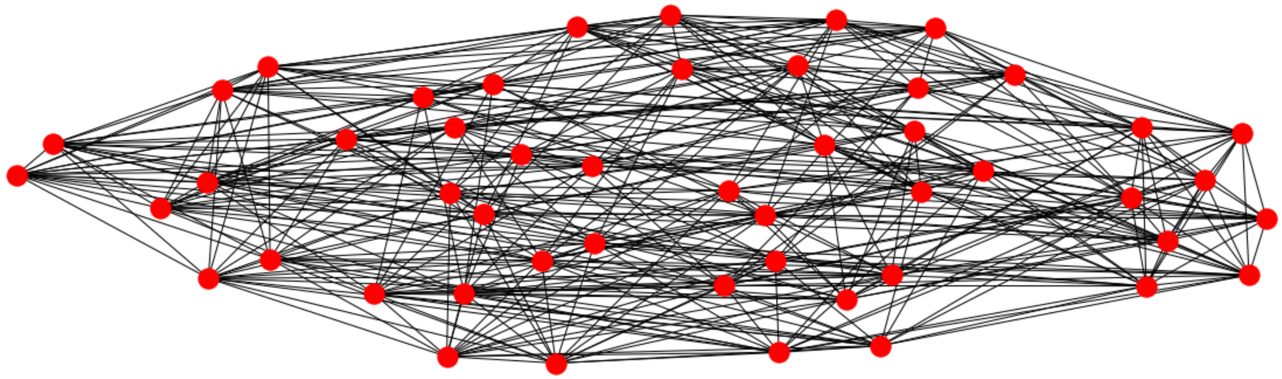
```
small_knn3d = graph.knn3d(graph.grid_coordinates(5), k=8, metric='cityblock', d=2)
nx.draw(small_knn3d)
```



3D wrap-around grid

In [44]:

```
small_wraps3d = graph.kwraps3d(5, kd=1, d=2)
nx.draw(small_wraps3d)
```

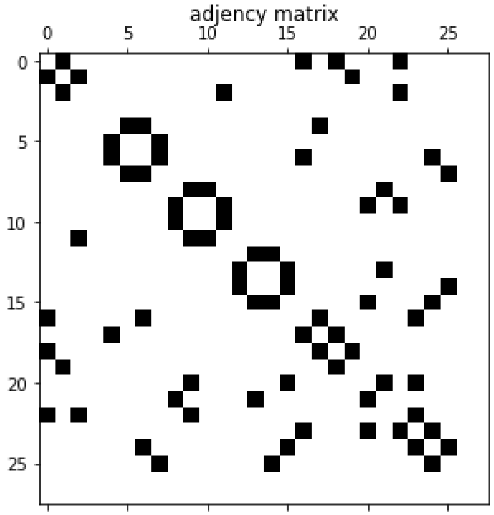


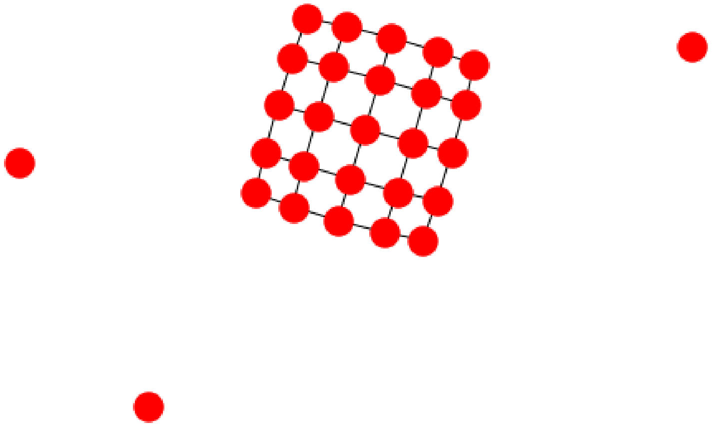
Coarsening

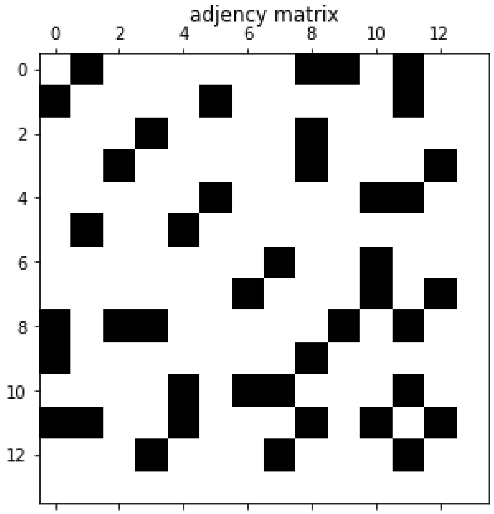
Graclus

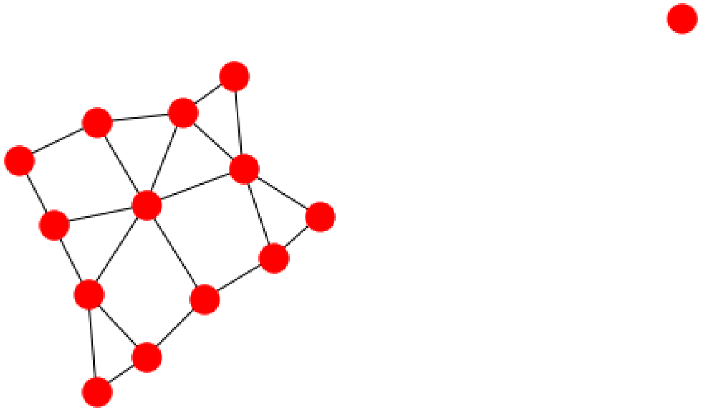
In [50]:

```
viz.plot_graph_steps(graclus_levels)
```





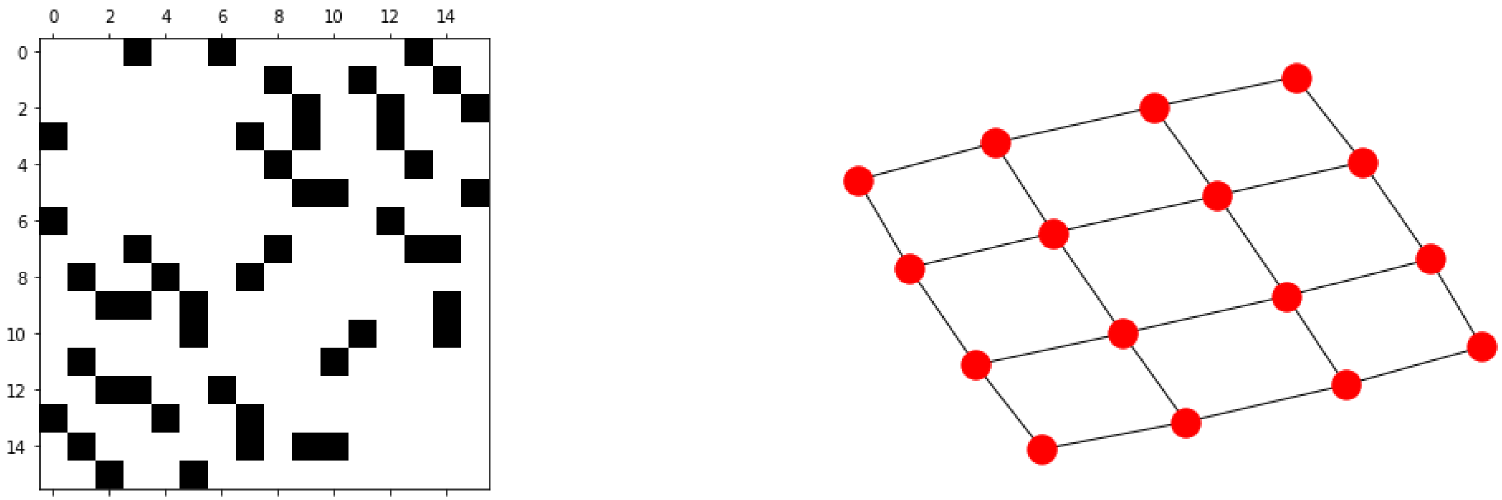


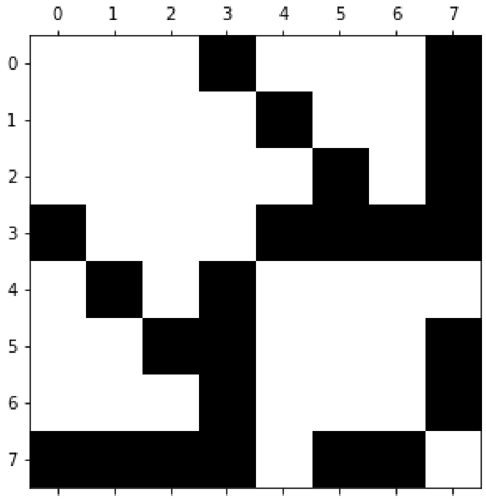


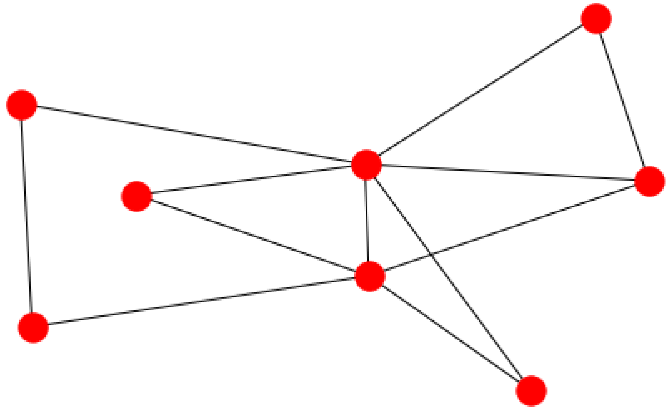
Algebraic multigrid

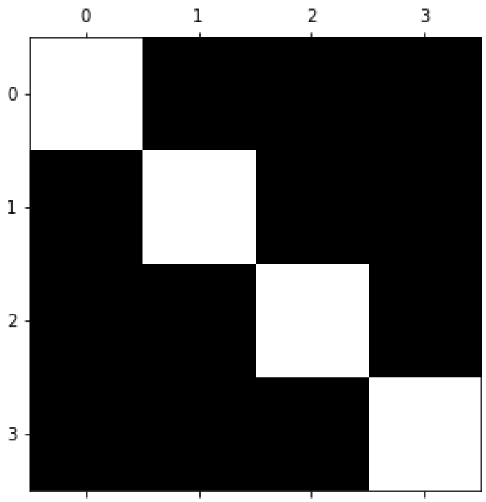
In [53]:

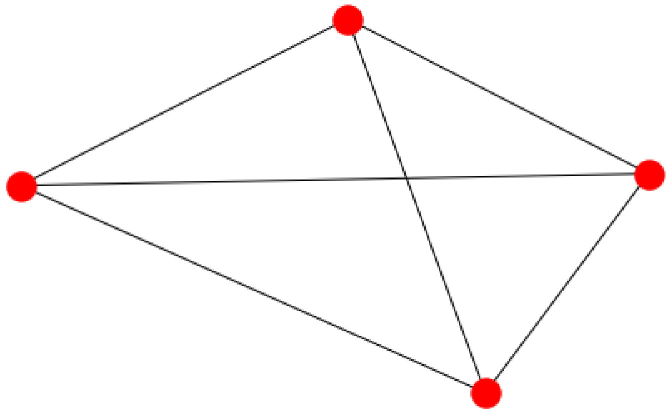
```
for g in graphs:
    plt.subplot(121)
    plt.spy(g.todense())
    plt.subplot(122)
    nx.draw(nx.from_numpy_array(g.todense()))
    plt.show()
```




0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7



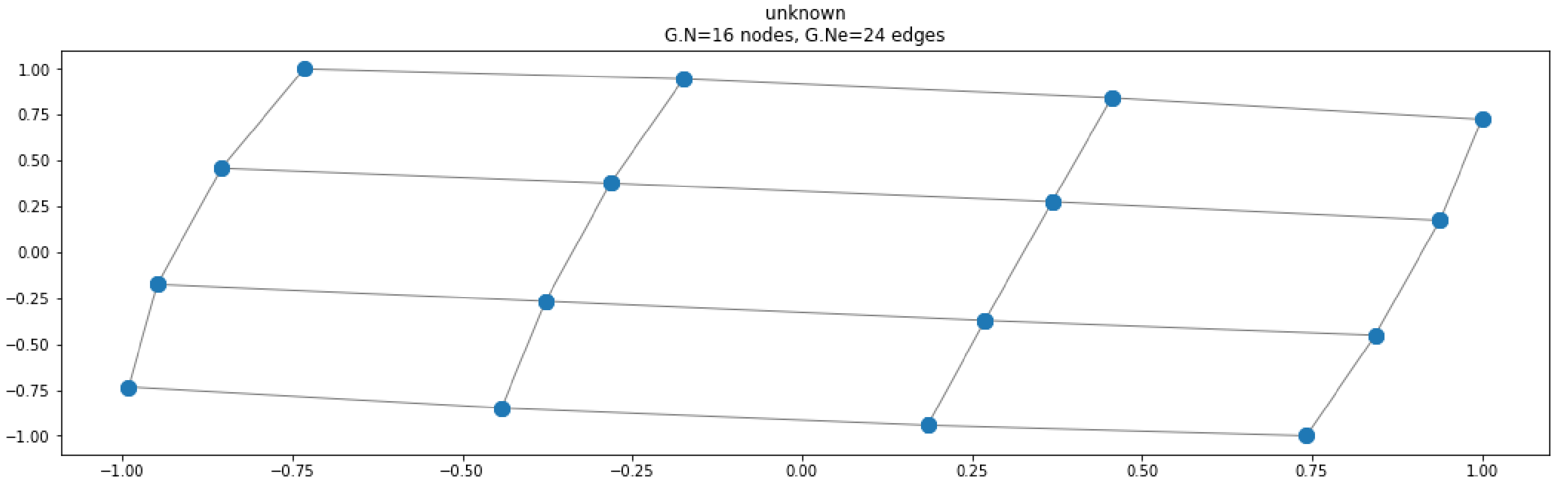

0 1 2 3
0 1 2 3

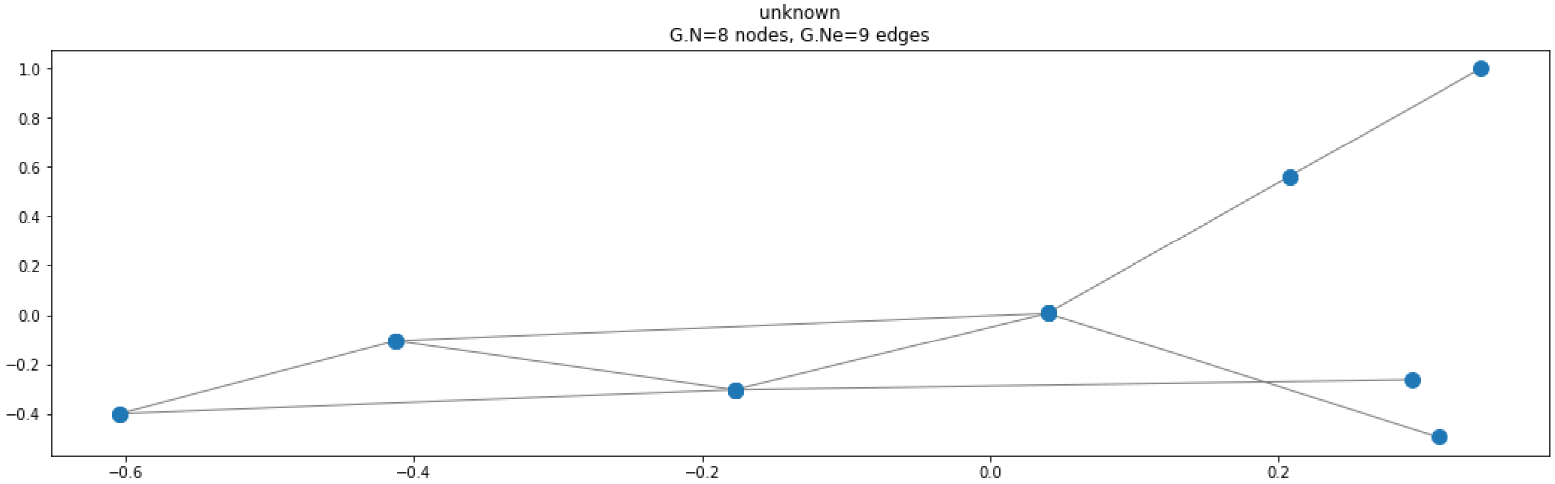


Kron reduction

In [56]:

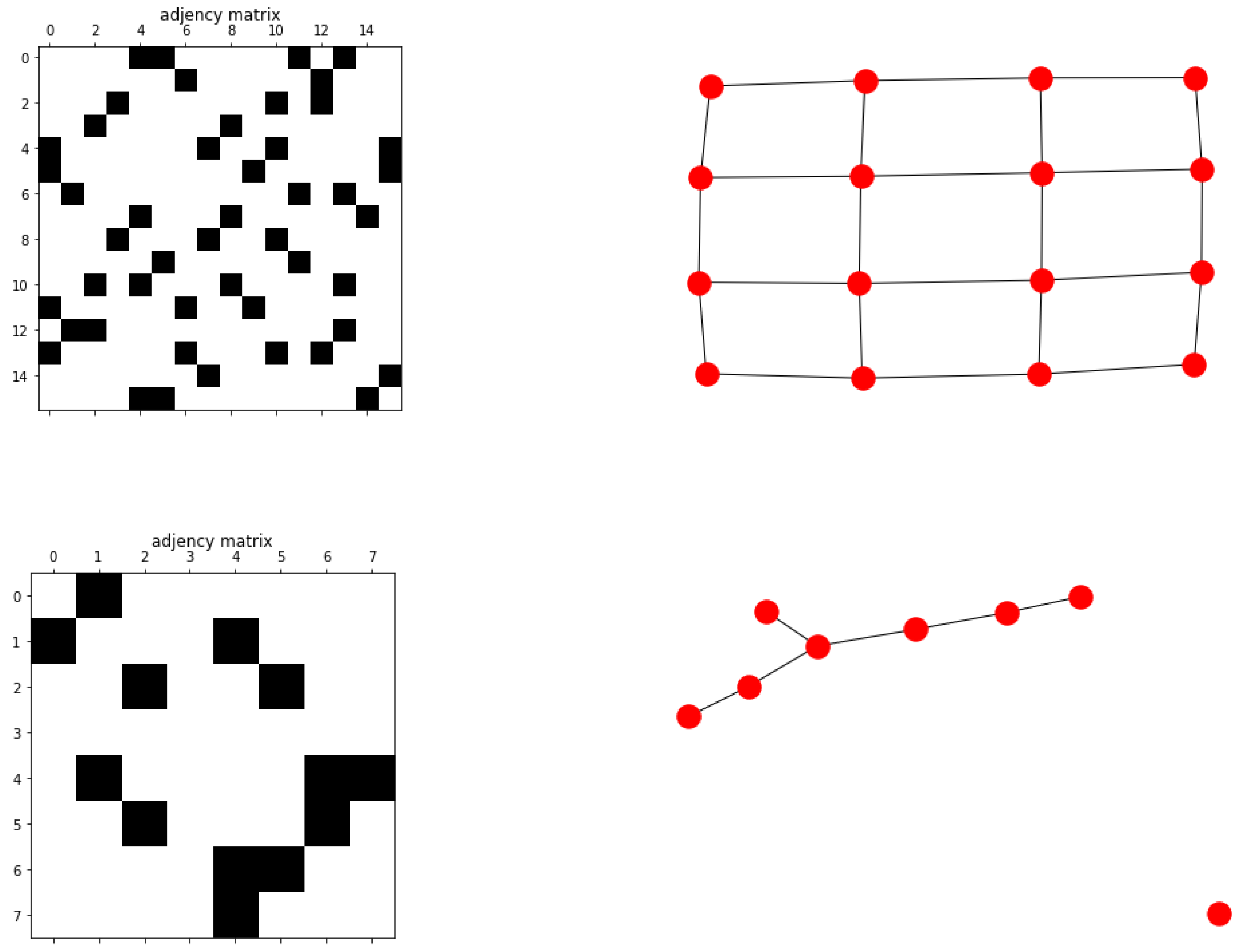
```
for g in kron_levels:
    g.set_coordinates()
    g.plot()
```





Maximum spanning tree

In [59]:
```
viz.plot_graph_steps(mst_levels)
```



5 - Models

Convolution baseline

Layer	Output shape	Input connected to
Convolution2d	16	Convolution2d_input
MaxPool2d	4	Convolution2d
Convolution2d	32	MaxPool2d
MaxPool2d	4	Convolution2d
Linear	256	MaxPool2d
Linear	128	Linear
Linear	1	Linear

Graph convolution (Graclus)

Layer	Output shape	Input connected to
GraphFourierConv / GraphChebyshevConv	16	GraphFourierConv_input / GraphChebyshevConv_input
MaxPool2d	4	GraphFourierConv / GraphChebyshevConv
GraphFourierConv / GraphChebyshevConv	32	MaxPool2d
MaxPool2d	4	GraphFourierConv / GraphChebyshevConv
Linear	256	MaxPool2d
Linear	128	Linear
Linear	1	Linear

6 - Evaluation

In [99]:

```
scores
```

Out[99]:

	accuracy	precision	recall	f1
name				
baseline	0.502075	0.470085	0.486726	0.478261
knn	0.684647	0.645669	0.725664	0.683333
logistic	0.775934	0.725191	0.840708	0.778689
conv	0.854772	0.848214	0.840708	0.844444
gcnn_grid	0.788382	0.822917	0.699115	0.755981
gcnn_kwraps	0.780083	0.705479	0.911504	0.795367

7 - Conclusion

Possible improvements

- sparse operations
- better and more detailed comparison
- speedup to improve both coarsening algorithms and training
- graph deconvolution and inspect what the filter actually does

8 - References

- TORRES, Ramon, SNOEIJ, Paul, GEUDTNER, Dirk, et al. GMES Sentinel-1 mission. Remote Sensing of Environment, 2012, vol. 120, p. 9-24.
- SHUMAN, David I., NARANG, Sunil K., FROSSARD, Pascal, et al. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine, 2013, vol. 30, no 3, p. 83-98.
- BRONSTEIN, Michael M., BRUNA, Joan, LECUN, Yann, et al. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine, 2017, vol. 34, no 4, p. 18-42.
- DEFFERRARD, Michaël, BRESSON, Xavier, et VANDERGHEYNST, Pierre. Convolutional neural networks on graphs with fast localized spectral filtering. In : Advances in Neural Information Processing Systems. 2016. p. 3844-3852.
- NGUYEN Ha Q., DO Minh N, et al. Downsampling of Signal on Graphs Via Maximum Spanning Trees. IEEE Transactions on Signal Processing, 2015, vol. 63, no 1.
- DORFLER Florain, BULLO Francesco. Kron reduction of graphs with applications to electrical networks. 2011.