Maximal weight matching

Let us take matchings one step further. Given a graph G, let $w : E(G) \to \mathbb{R}$ be edge weight 5^5
If M is a matching in G, then its weight, $w(G)$, is defined as the sum of weight of its edges

$$w(M) = \sum_{e \in M} w(e)$$

The MaximalWeightMatching problem can be defined as

input: Graph G, edge weights w.

output: Matching M in G for which $w(M)$ is maximal.

Let G_{21} be a 21×21 grid graph - the vertex set consists of integral points in the plane with coordinates between 0 and 20, vertices at distance 1 being adjacent. Choose and fix a choice of random edge weights

$$w : E\left(G_{21}\right) \to [1, 2]$$

11. Solve the MaximalWeightMatching problem on G_{21}, w using a freely available LP solver - you should be solving a linearly relaxed version. Use both simplex and interior point method approaches. Is your solution a matching?

In [ ]:

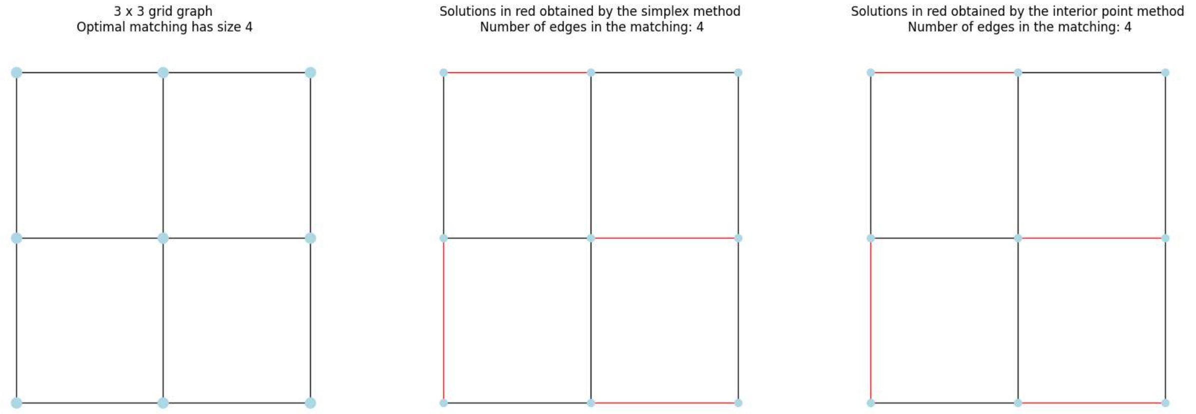
```
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
from scipy.optimize import linprog
import warnings
warnings.filterwarnings('ignore')

def plot_maximal_weight_matching(n=21, weights=True, maxiter=None):
    fig, ax = plt.subplots(1, 3, figsize=(21, 7))
    G = nx.grid_graph([n, n])
    pos = dict(zip(G,G))
    nx.draw(G, pos, node_color='lightblue', node_size=100, ax=ax[0])
    G_indexed = nx.convert_node_labels_to_integers(G, first_label=0, ordering='defau
    ax[0].set_title(f'{n} x {n} grid graph\n Optimal matching has size {(1+n)*(n//2)

    if weights: weights = np.random.rand(G.number_of_edges()) + 1
    else: weights = None
    res = maximal_weight_matching(G_indexed, weights=weights, method='simplex', maxi
    edge_colors = {edge: 'red' if res[i] > 10e-8 else 'black' for i, edge in enumera
    nx.draw(G, pos, node_color='lightblue', node_size=50, edge_color=[edge_colors[ed
    ax[1].set_title(f'Solutions in red obtained by the simplex method\n Number of ed

    res = maximal_weight_matching(G_indexed, weights=weights, method='interior-point
    edge_colors = {edge: 'red' if res[i] > 10e-8 else 'black' for i, edge in enumera
    nx.draw(G, pos, node_color='lightblue', node_size=50, edge_color=[edge_colors[ed
    ax[2].set_title(f'Solutions in red obtained by the interior point method\n Numbe

def maximal_weight_matching(G, weights=None, method='highs', maxiter=None):
    dic = {edge: i for i,edge in enumerate(G.edges())}

    # Define the matrix A and the vector b
    A = np.zeros((G.number_of_nodes(), G.number_of_edges()))
```

```
    for node in G.nodes():
        for edge in G.edges(node):
            if edge in dic:
                A[node][dic[edge]] = 1
            if (edge[1], edge[0]) in dic:
                A[node][dic[(edge[1], edge[0])]] = 1

    b = np.ones(G.number_of_nodes())

    # Define the bounds for the variables
    bounds = [(0, 1) for i in range(G.number_of_edges())]

    if weights is not None:
        # Use weight matching
        c = -weights
    else:
        # We are maximizing, but linprog minimizes, so we multiply by -1
        c = -np.ones(G.number_of_edges())

    # Solve the linear program
    solution = linprog(c, A_ub=A, b_ub=b, bounds=bounds, method=method, options=None
    print(f'INFO for {method} method')
    print(f'Number of iterations: {solution.nit}')
    print('Sum of solution: ', sum(solution.x))
    return solution.x
```

In [ ]:

```
plot_maximal_weight_matching(n=3, weights=True)
```

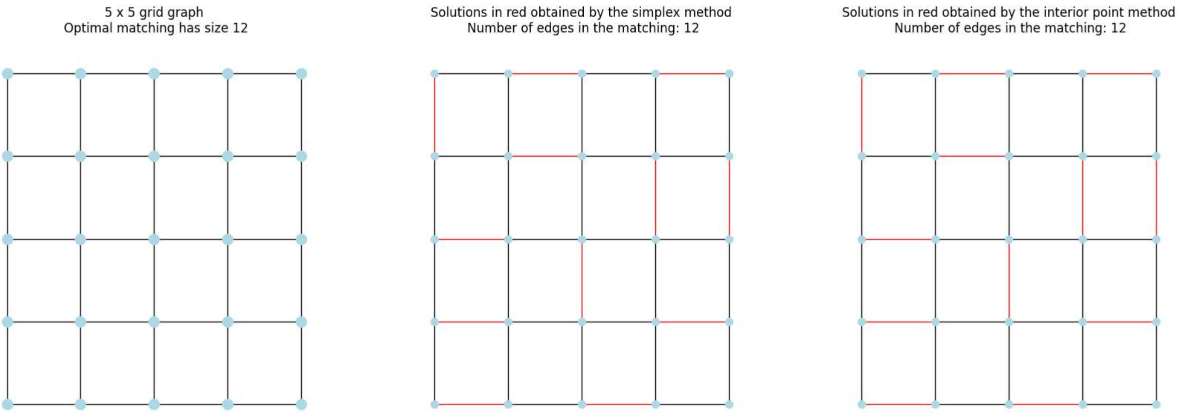
```
INFO for simplex method
Number of iterations: 36
Sum of solution:  4.0
INFO for interior-point method
Number of iterations: 6
Sum of solution:  4.000000003994763
```



In [ ]:

```
plot_maximal_weight_matching(n=5, weights=True)
```

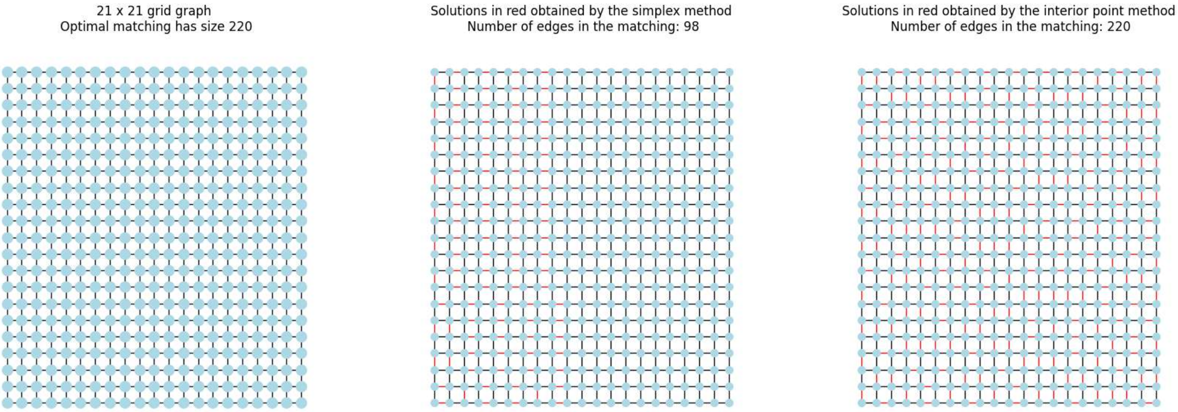
```
INFO for simplex method
Number of iterations: 156
Sum of solution:  12.0
INFO for interior-point method
Number of iterations: 7
Sum of solution:  12.00000000000938
```



In [ ]:

```
plot_maximal_weight_matching(n=21, weights=True)
```

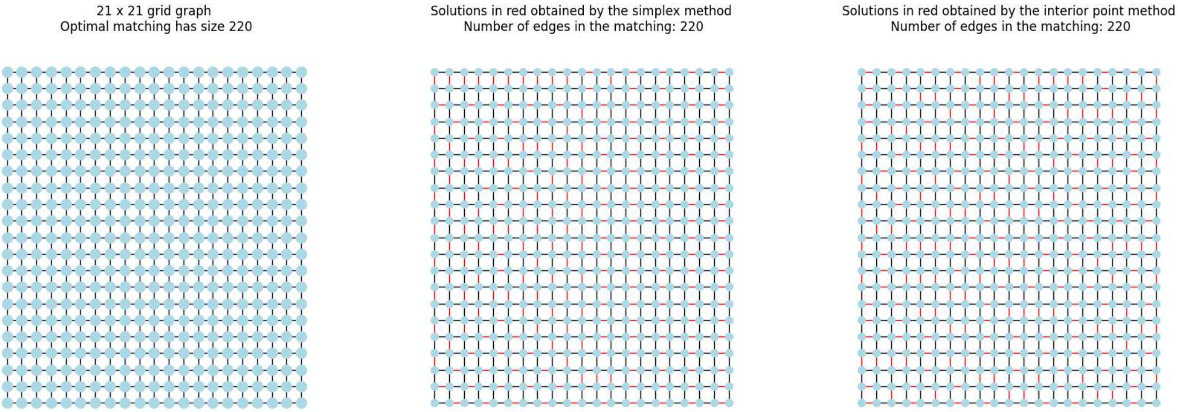
```
INFO for simplex method
Number of iterations: 1000
Sum of solution:  98.0
INFO for interior-point method
Number of iterations: 9
Sum of solution:  220.00000000120062
```



In [ ]:

```
plot_maximal_weight_matching(n=21, weights=True, maxiter=5000)
```

```
INFO for simplex method
Number of iterations: 5000
Sum of solution:  220.0
INFO for interior-point method
Number of iterations: 8
Sum of solution:  220.00000000559066
```



We again show our results using plots. We can see that simplex method needs a lot more iterations for any graph size. If we use the default number of iterations (1000) the simplex method finds a matching but not the optimal one. Changing the maxiter parameter to 5000 allows the simplex method to find the optimal matching for the 21x21 grid graph. We can also see that sum of solution for the interior point method is not an integer, which means that the solution we obtain is a matching but due to numerical issues we do not obtain exact integers.