

In [48]:

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import os
6 from IPython.display import display
7
8 sns.set(font_scale=1.5)
```

In [294]:

```
1 def parse_file(df, path, group, file):
2     full_path = path + '/' + file
3     known_lb = 0
4     known_ub = 0
5     negative_weight = 0
6     collected = {}
7     with open(full_path, 'r') as f:
8         for line in f:
9             key, value = line.split(':')
10
11             if key.startswith('Known lower'):
12                 known_lb = int(value)
13                 continue
14             if key.startswith('Known upper'):
15                 known_ub = int(value)
16                 continue
17             if key.startswith('Negative'):
18                 negative_weight = int(value)
19                 continue
20
21             algo = ''
22             if key.startswith('Brute'):
23                 algo = 'bruteforce'
24             elif key.startswith('Rand'):
25                 algo = 'random'
26             elif key.startswith('Greed'):
27                 algo = 'greed'
28             elif key.startswith('Goemans'):
29                 algo = 'goemans-williamson'
30
31             if algo not in collected:
32                 collected[algo] = [0, 0]
33
34             if key.endswith('(answer)'):
35                 collected[algo][0] = int(value)
36                 known_lb = max(known_lb, int(value))
37             else:
38                 collected[algo][1] = float(value)
39         for algo, (ans, time) in collected.items():
40             group_name = group
41             if 'biqmac' in group:
42                 group_name = group + '/' + file[:-2]
43     df.loc[len(df)] = [
```

```

44         group_name, algo, ans, time, known_lb, known_ub, ne
45     ]
46
47
48 def parse_results(path='../testing/results'):
49     df = pd.DataFrame(columns=[
50         'group', 'algo', 'weight', 'time', 'known_lower_bound',
51         'known_upper_bound', 'negative_weight', 'filename'
52     ])
53     for root, dirs, files in os.walk(path):
54         if not files:
55             continue
56         dir_name = root.split('results/')[-1]
57         for file in files:
58             parse_file(df, root, dir_name, file)
59     return df
60
61
62 def add_solutions(df, solutions_file):
63     with open(solutions_file, 'r') as f:
64         for line in f:
65             line = line.strip()
66             if line != '':
67                 file, ans = line.split(' ')
68                 ans = int(ans)
69                 ind = df[df['filename'] == file].index
70                 df.loc[ind, 'known_lower_bound'] = ans

```

In [340]:

```

1 def accumulate_results(all_results):
2     df = pd.DataFrame(columns=[
3         'group', 'algo', 'mean_time', 'mean_accuracy_lb', 'mean
4     ])
5     for group_name, group_results in all_results.groupby('group'):
6         for algo_name, table in group_results.groupby('algo'):
7             df.loc[len(df)] = [
8                 group_name, algo_name,
9                 table.time.mean(),
10                table.accuracy_lb.mean(),
11                table.accuracy_ub.mean()
12            ]
13     return df

```

```
In [337]: 1 all_results = parse_results()
          2 all_results.head()
```

Out[337]:

	group	algo	weight	time	known_lower_bound	known_upper_bound
0	random_graph_50/size-8	bruteforce	8	0.00044	8	
1	random_graph_50/size-8	random	8	0.00007	8	
2	random_graph_50/size-8	greed	7	0.00000	8	
3	random_graph_50/size-8	goemans-williamson	7	0.02957	8	
4	random_graph_50/size-8	bruteforce	14	0.00049	14	

```
In [338]: 1 add_solutions(all_results, '../testing/tests/biqmac/solutions/i
          2 add_solutions(all_results, '../testing/tests/biqmac/solutions/r
```

```
In [339]: 1 all_results['accuracy_lb'] = \
          2     (all_results['weight'] - all_results['negative_weight']) /
          3     (all_results['known_lower_bound'] - all_results['negative_weight'])
          4 all_results['accuracy_ub'] = \
          5     all_results['weight'] / all_results['known_upper_bound']
```

```
In [341]: 1 results = accumulate_results(all_results)
          2 results.head()
```

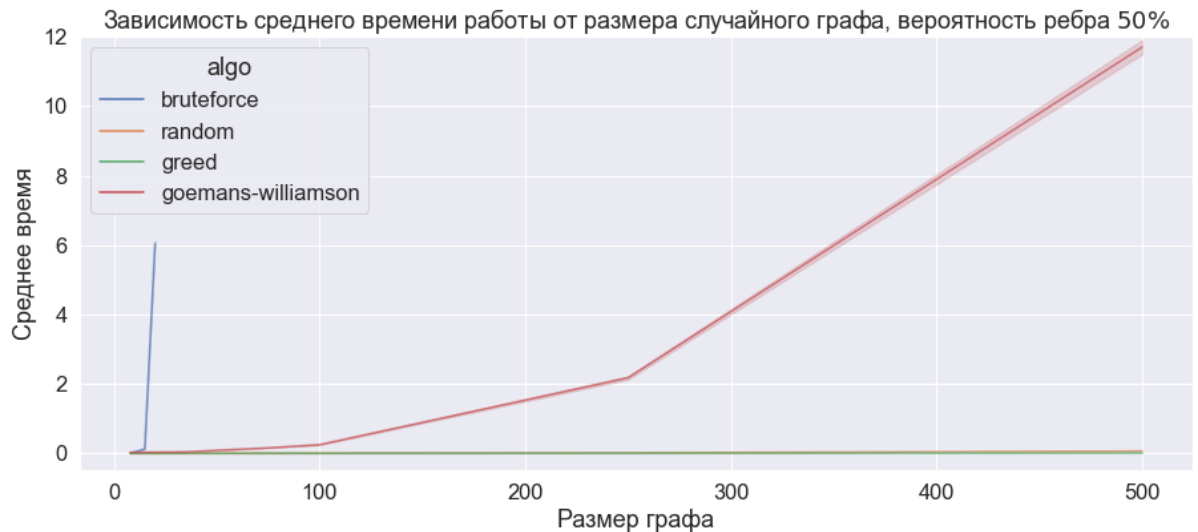
Out[341]:

	group	algo	mean_time	mean_accuracy_lb	mean_accuracy_ub
0	biqmac/ising/ising2.5-100_5555	goemans-williamson	0.27099	0.993725	0.354846
1	biqmac/ising/ising2.5-100_5555	greed	0.00061	0.993900	0.355094
2	biqmac/ising/ising2.5-100_6666	goemans-williamson	0.39158	0.993726	0.344382
3	biqmac/ising/ising2.5-100_6666	greed	0.00053	0.995010	0.346923
4	biqmac/ising/ising2.5-100_7777	goemans-williamson	0.35093	0.977962	0.379060

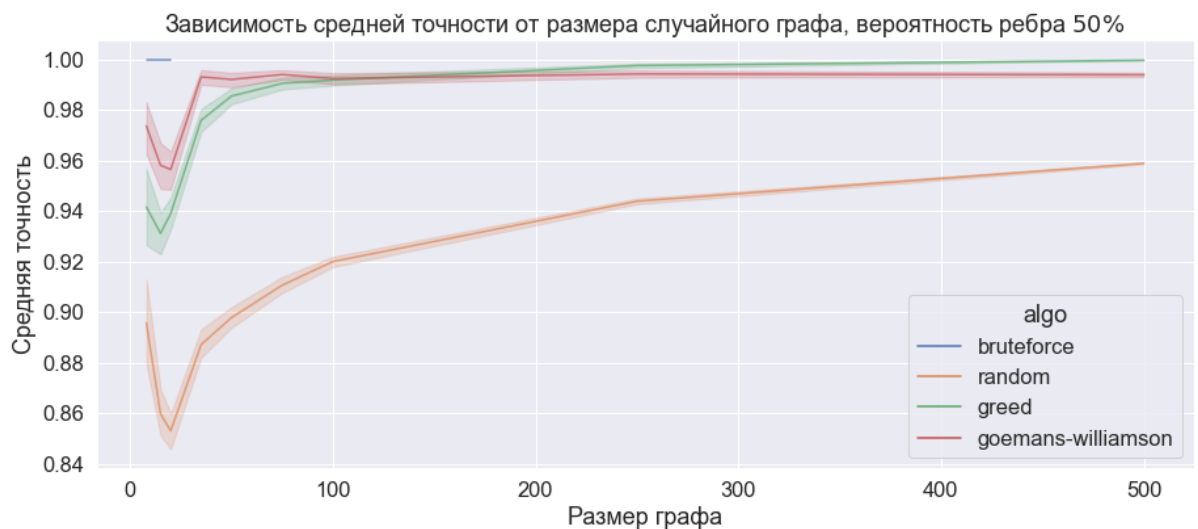
```
In [329]: 1 def plot_stats_in_group(df, name_filter, param, title=None,
2             xlabel=None, ylabel=None, xlim=None, ylim=None,
3             add_theoretical_est=False):
4     with sns.axes_style('darkgrid'):
5         plt.figure(figsize=(15, 6))
6         sns.lineplot(data=df,
7                     x=df['group'].apply(name_filter),
8                     y=df[param].astype(np.float),
9                     hue='algo',
10                    alpha=0.7)
11     if add_theoretical_est:
12         xmin = df['group'].apply(name_filter).min()
13         xmax = df['group'].apply(name_filter).max()
14         plt.hlines(0.87856, xmin=xmin, xmax=xmax,
15                  linestyle='--', color='black',
16                  alpha=0.7,
17                  label=r'Теоретическая оценка $88\%$')
18         plt.legend()
19     plt.xlim(xlim)
20     plt.ylim(ylim)
21     if title is not None:
22         plt.title(title)
23     if xlabel is not None:
24         plt.xlabel(xlabel)
25     if ylabel is not None:
26         plt.ylabel(ylabel)
```

```
In [216]: 1 def plot_accuracy(df, name_filter, title=None,
2           xlabel=None, ylabel=None, xlim=None, ylim=None):
3     with sns.axes_style('whitegrid'):
4         plt.figure(figsize=(15, 10))
5         all_x = df['group'].apply(name_filter)
6         x_min = all_x.min()
7         x_max = all_x.max()
8         for algo in df.algo.unique():
9             data = []
10            x = df[df['algo'] == algo]['group'].apply(name_filter)
11            y_min = df[df['algo'] == algo]['mean_accuracy_lb']
12            y_max = df[df['algo'] == algo]['mean_accuracy_ub']
13            data = list(zip(x, y_min, y_max))
14            data.sort(key=lambda x: x[0])
15            x = list([d[0] for d in data])
16            y_min = list([d[1] for d in data])
17            y_max = list([d[2] for d in data])
18
19            plt.fill_between(x, y_min, y_max, label=algo, alpha=0.5)
20
21            plt.hlines(0.87856, xmin=x_min, xmax=x_max, linestyle='dashed',
22                      label=r'Теоретическая оценка $88\%$')
23
24
25            plt.xlim(xlim)
26            plt.ylim(ylim)
27
28            plt.legend(loc='lower left')
29            if title is not None:
30                plt.title(title)
31            if xlabel is not None:
32                plt.xlabel(xlabel)
33            if ylabel is not None:
34                plt.ylabel(ylabel)
```

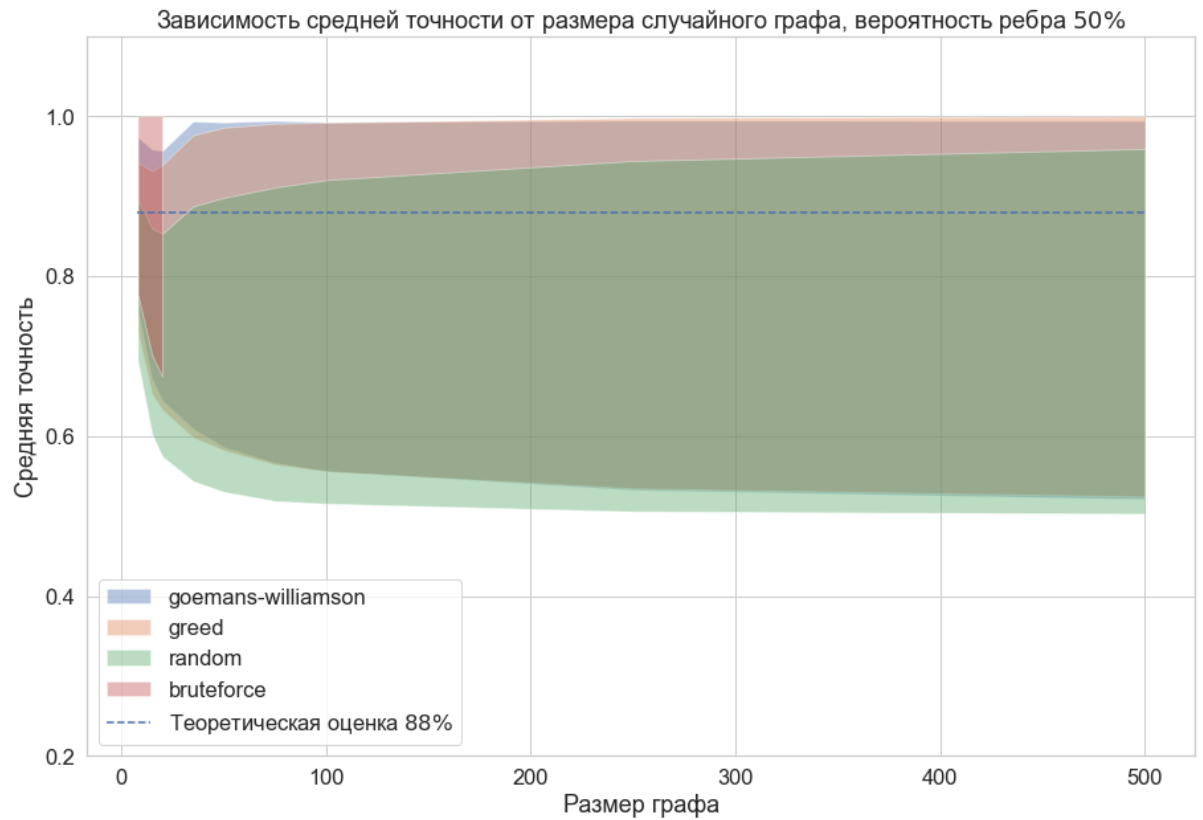
```
In [277]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2             lambda x: int(x.split('/')[1].split('-')[-1]
3             'time',
4             title='Зависимость среднего времени работы
5             'от размера случайного графа, '
6             r'вероятность ребра $50\%$',
7             xlabel='Размер графа',
8             ylabel='Среднее время',
9             ylim=(-0.5, 12))
```



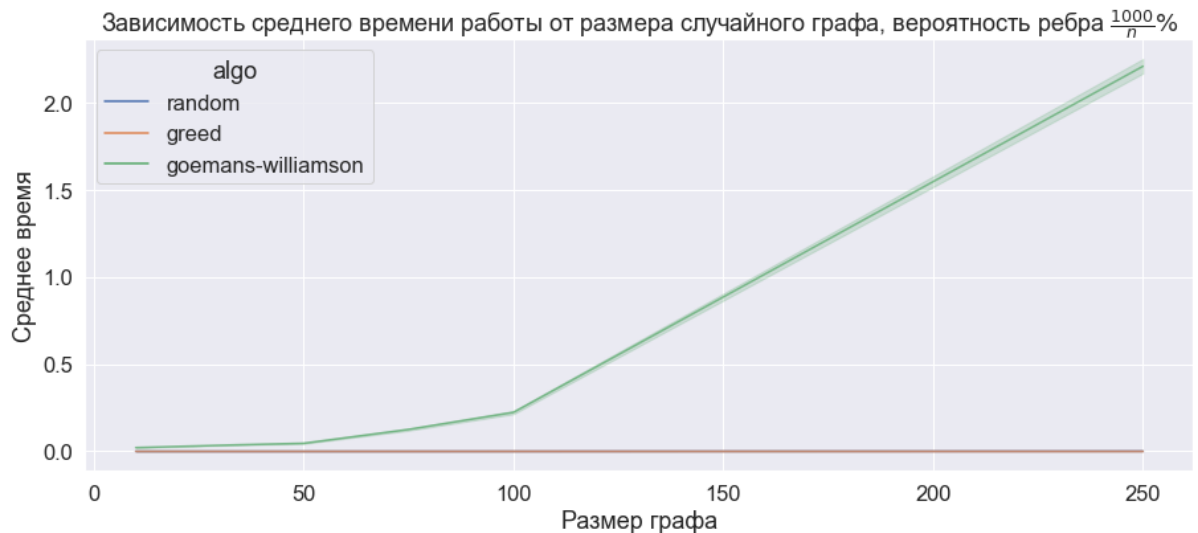
```
In [276]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2             lambda x: int(x.split('/')[1].split('-')[-1]
3             'accuracy_lb',
4             title='Зависимость средней точности '
5             'от размера случайного графа, '
6             r'вероятность ребра $50\%$',
7             xlabel='Размер графа',
8             ylabel='Средняя точность')
```



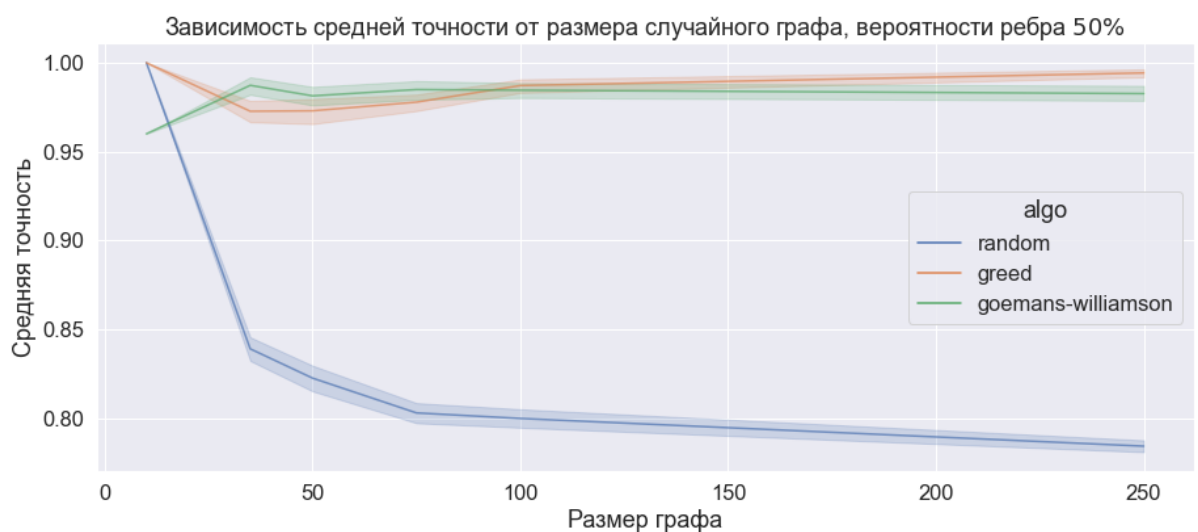
```
In [279]: 1 plot_accuracy(results[results['group'].str.contains('random_gra
2         lambda x: int(x.split('/')[1].split('-')[1]
3         title='Зависимость средней точности '
4         'от размера случайного графа, '
5         r'вероятность ребра $50\%$',
6         xlabel='Размер графа',
7         ylabel='Средняя точность',
8         ylim=(0.2, 1.1))
```



```
In [281]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[1].split('-')[1]
3         'time',
4         title='Зависимость среднего времени работы
5         'от размера случайного графа, '
6         r'вероятности ребра $\frac{1000}{n}$%',
7         xlabel='Размер графа',
8         ylabel='Среднее время')
```

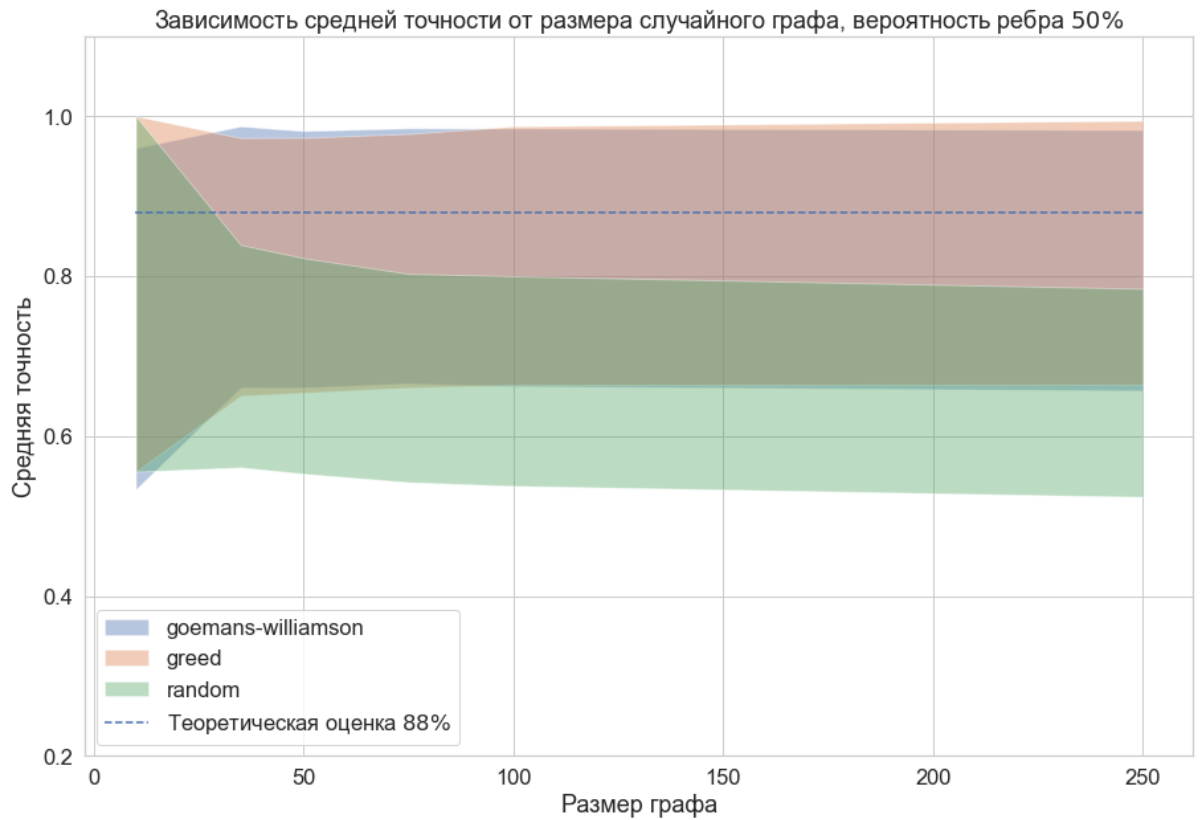


```
In [286]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[1].split('-')[1]
3         'accuracy_lb',
4         title='Зависимость средней точности '
5         'от размера случайного графа, '
6         r'вероятности ребра $50\%$',
7         xlabel='Размер графа',
8         ylabel='Средняя точность')
```





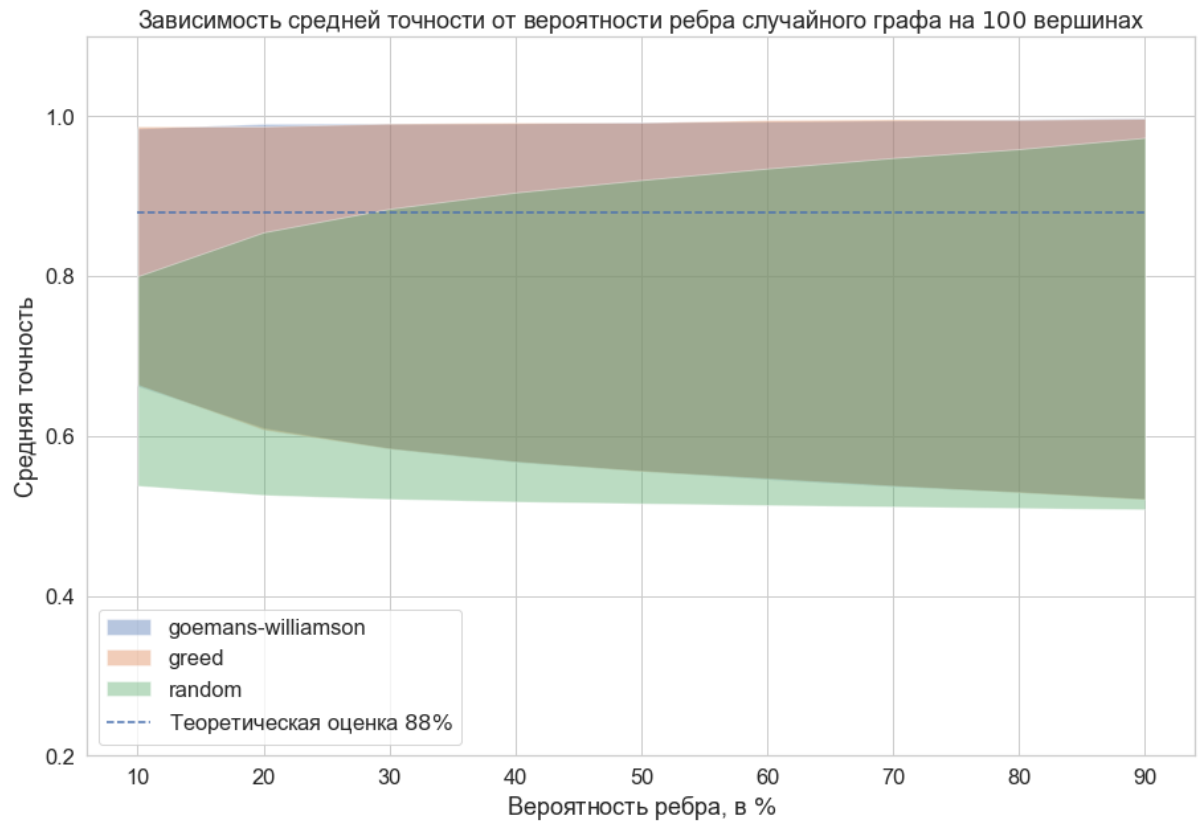
```
In [219]: 1 plot_accuracy(results[results['group'].str.contains('random_gra
2           lambda x: int(x.split('/')[1].split('-')[1]
3           title='Зависимость средней точности '
4           'от размера случайного графа, '
5           r'вероятность ребра $50\%$',
6           xlabel='Размер графа',
7           ylabel='Средняя точность',
8           ylim=(0.2, 1.1))
```



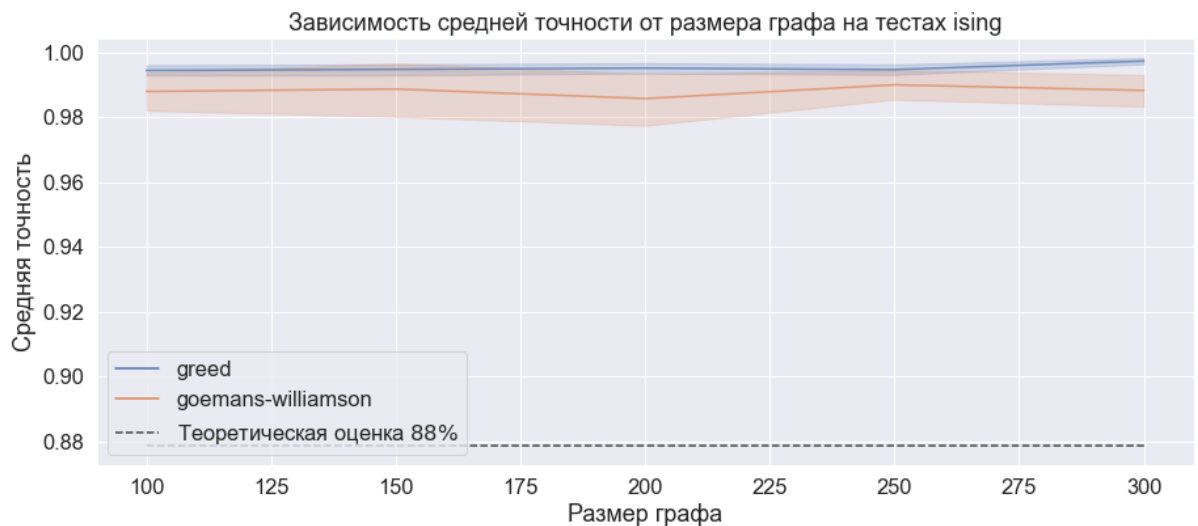
```
In [284]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2             lambda x: int(x.split('/')[1]).split('-')[1]
3             'accuracy_lb',
4             title='Зависимость средней точности '
5             'от вероятности ребра случайного графа '
6             r'на $100$ вершинах',
7             xlabel=r'Вероятность ребра, в $\\%$',
8             ylabel='Средняя точность')
```



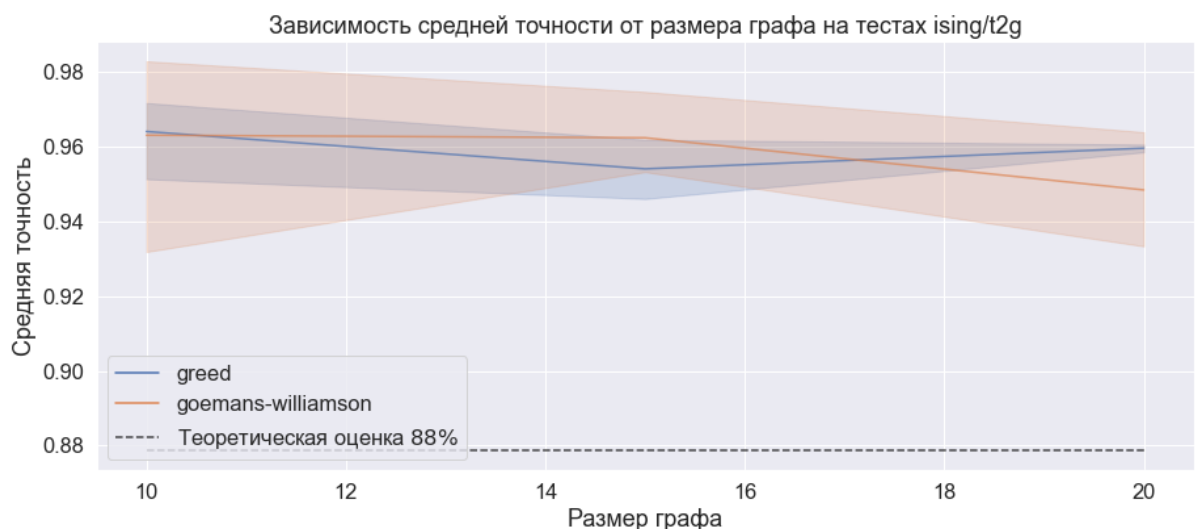
```
In [285]: 1 plot_accuracy(results[results['group'].str.contains('random_gra
2          lambda x: int(x.split('/')[1].split('-')[1]
3          title='Зависимость средней точности '
4          'от вероятности ребра случайного графа '
5          r'на $100$ вершинах',
6          xlabel=r'Вероятность ребра, в $\%$',
7          ylabel='Средняя точность',
8          ylim=(0.2, 1.1))
```



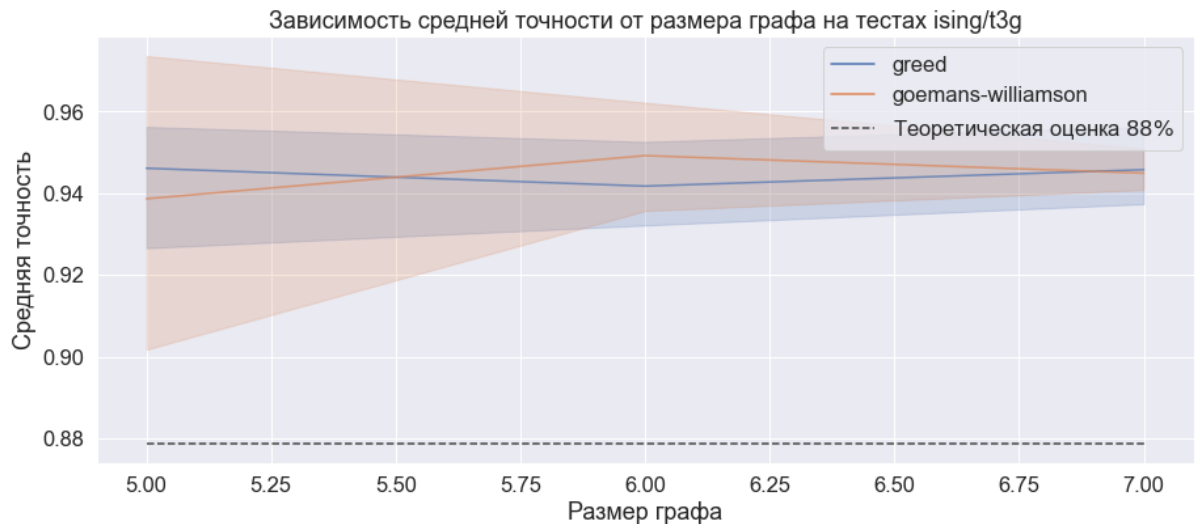
```
In [330]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[1].split('-')[-1]
3         'accuracy_lb',
4         title='Зависимость средней точности '
5         'от размера графа '
6         r'на тестах ising',
7         xlabel=r'Размер графа',
8         ylabel='Средняя точность',
9         add_theoretical_est=True)
```



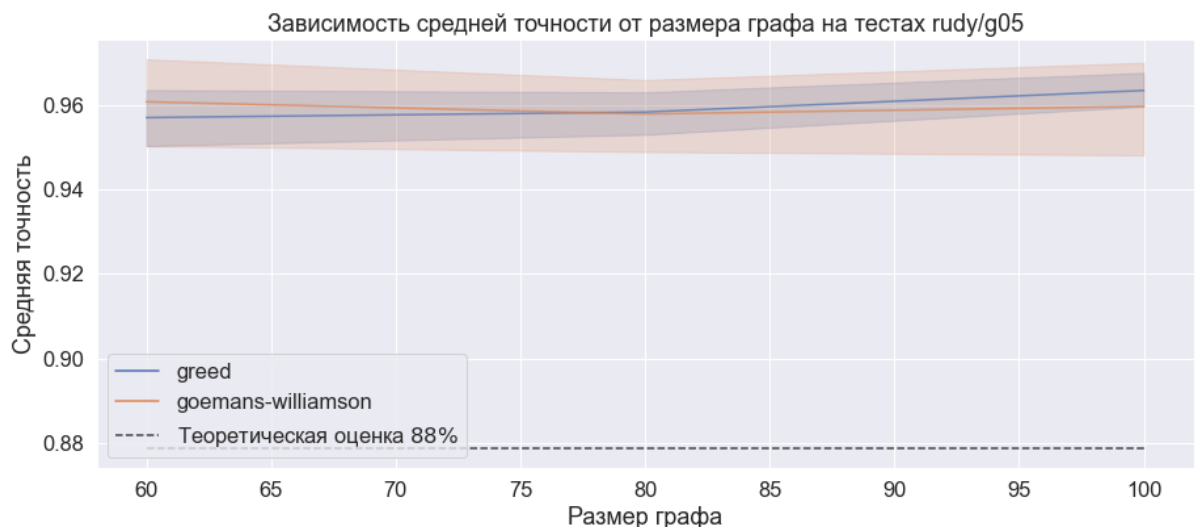
```
In [331]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[1].split('_')[0]
3         'accuracy_lb',
4         title='Зависимость средней точности '
5         'от размера графа '
6         r'на тестах ising/t2g',
7         xlabel=r'Размер графа',
8         ylabel='Средняя точность',
9         add_theoretical_est=True)
```



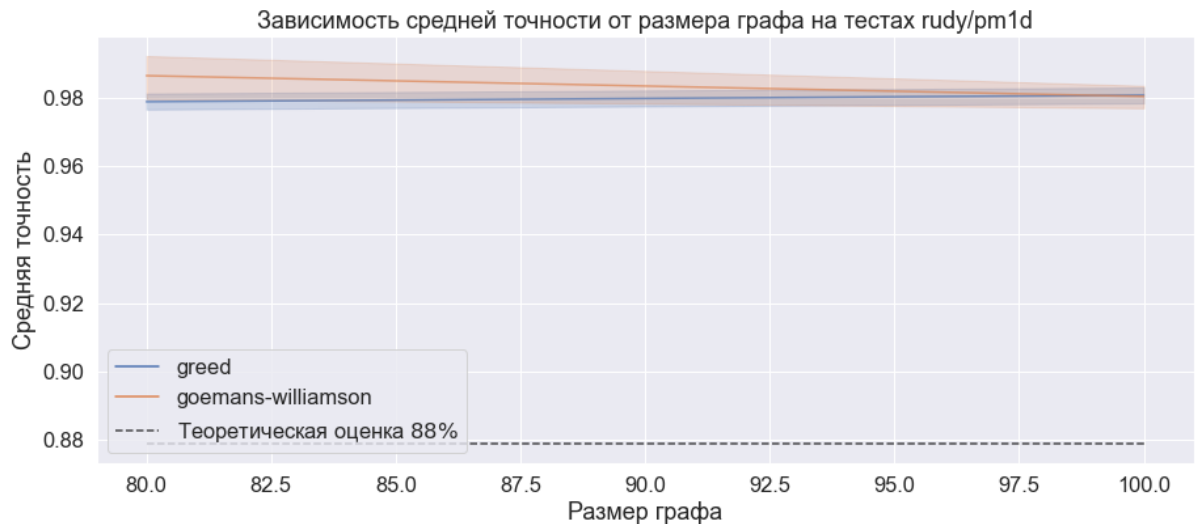
```
In [332]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[1].split('_')[0]
3         'accuracy_lb',
4         title='Зависимость средней точности '
5         'от размера графа '
6         r'на тестах ising/t3g',
7         xlabel=r'Размер графа',
8         ylabel='Средняя точность',
9         add_theoretical_est=True)
```



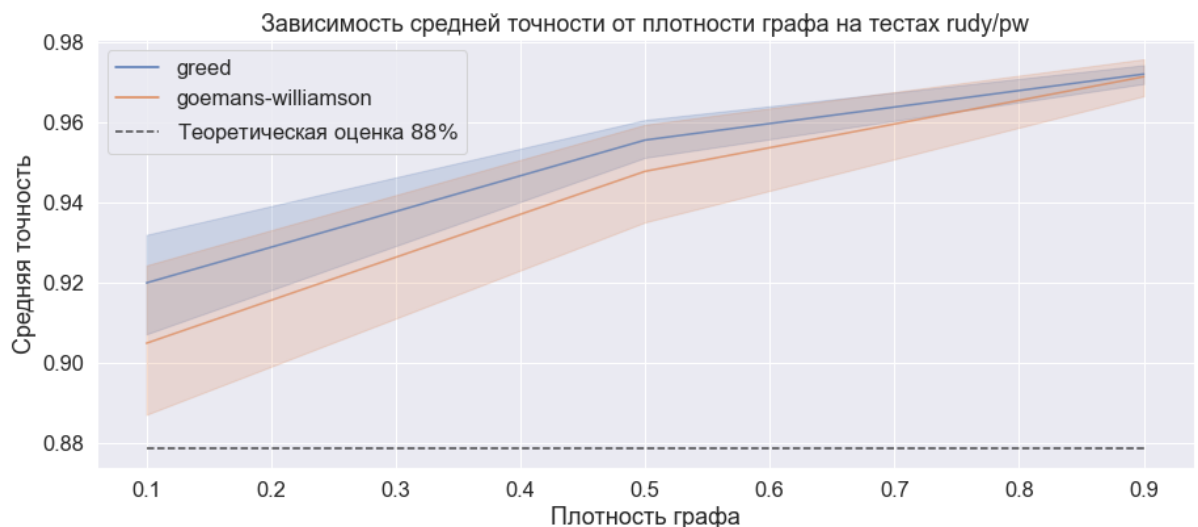
```
In [333]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[1].split('_')[1]
3         'accuracy_lb',
4         title='Зависимость средней точности '
5         'от размера графа '
6         r'на тестах rudy/g05',
7         xlabel=r'Размер графа',
8         ylabel='Средняя точность',
9         add_theoretical_est=True)
```



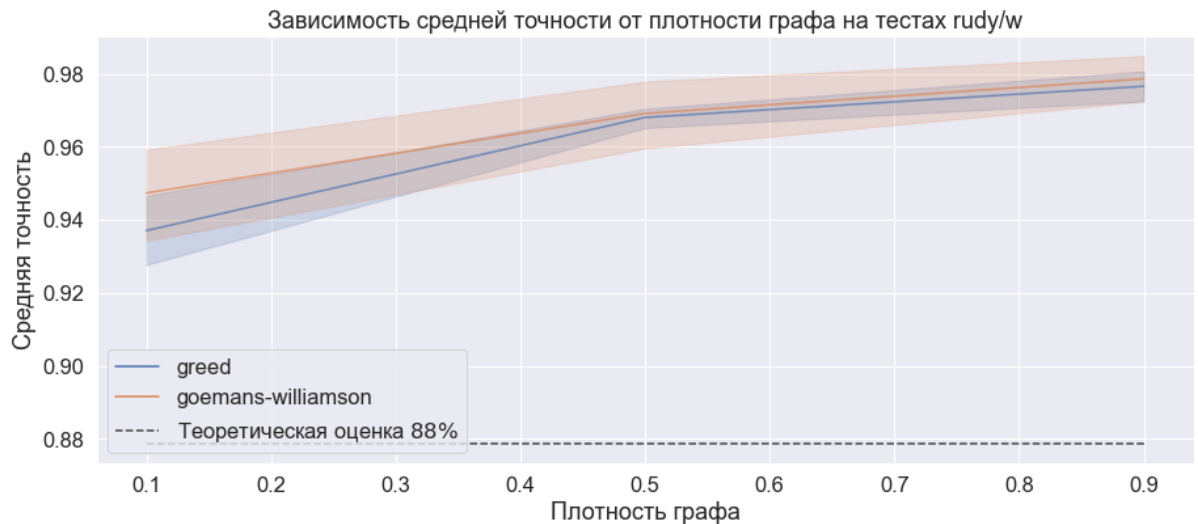
```
In [334]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[1].split('_')[1]
3         'accuracy_lb',
4         title='Зависимость средней точности '
5         'от размера графа '
6         r'на тестах rudy/pm1d',
7         xlabel=r'Размер графа',
8         ylabel='Средняя точность',
9         add_theoretical_est=True)
```



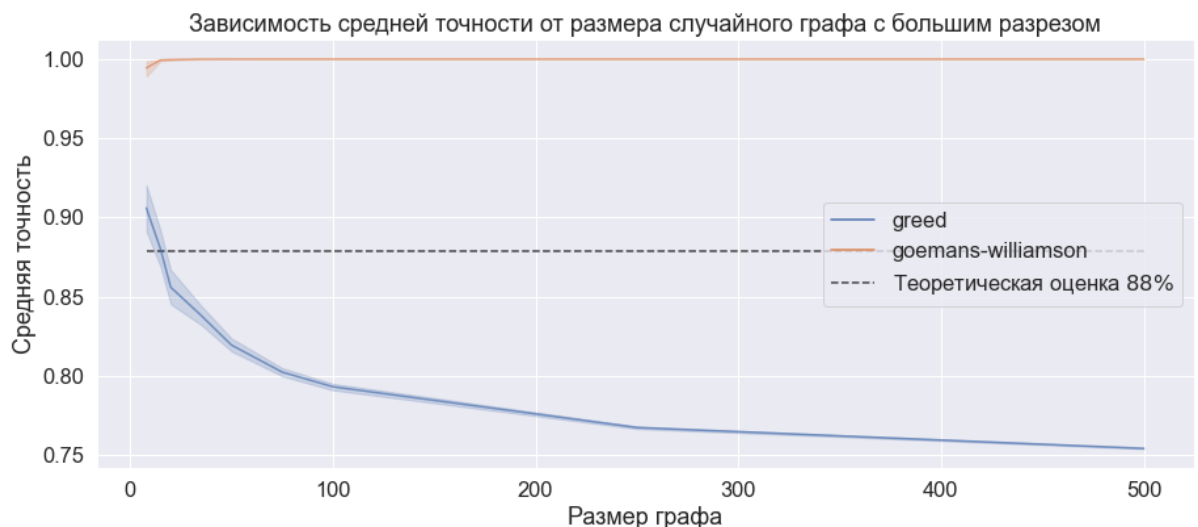
```
In [335]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[1].split('_')[0]
3         'accuracy_lb',
4         title='Зависимость средней точности '
5         'от плотности графа '
6         r'на тестах rudy/pw',
7         xlabel=r'Плотность графа',
8         ylabel='Средняя точность',
9         add_theoretical_est=True)
```



```
In [336]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[0]).split('_')[0] == 10000,
3         'accuracy_lb',
4         title='Зависимость средней точности '
5         'от плотности графа '
6         r'на тестах rudy/w',
7         xlabel=r'Плотность графа',
8         ylabel='Средняя точность',
9         add_theoretical_est=True)
```



```
In [343]: 1 plot_stats_in_group(all_results[all_results['group'].str.contains(
2         lambda x: int(x.split('/')[0]).split('_')[0] == 10000,
3         'accuracy_lb',
4         title='Зависимость средней точности '
5         'от размера случайного графа '
6         r'с большим разрезом',
7         xlabel='Размер графа',
8         ylabel='Средняя точность',
9         add_theoretical_est=True)
```



In [ ]:

1