current_optimum_bucket_pq_100_vs_hmetis

May 6, 2014

```
In [1]: import pandas as pd
        import pandas.io.sql as pd_sql
        import sqlite3 as sql
        import matplotlib.pyplot as plt

        #get data for kahypar
        kahypar_connection = sql.connect("/home/schlag/repo/schlag_git/benchmark/results/2014-05-05_cur
        kahypar_data = pd_sql.read_frame("select * from experiments",kahypar_connection)
        kahypar_min_cuts = pd.DataFrame(kahypar_data.groupby('graph')['cut'].min()).reset_index()

        #get data for hmetis
        hmetis_data = pd.read_csv('/home/schlag/repo/schlag_git/benchmark/results/2014-03-04_hmetis_rb/
        hmetis_min_cuts = pd.DataFrame(hmetis_data.groupby('graph')['cut'].min()).reset_index()

        # create dataframe with both min cuts and select the best
        both_min_cuts = pd.DataFrame(kahypar_min_cuts)
        both_min_cuts = both_min_cuts.rename(columns={'cut' : 'min_cut_kahypar'})
        both_min_cuts = pd.merge(both_min_cuts, hmetis_min_cuts, on='graph')
        both_min_cuts = both_min_cuts.rename(columns={'cut' : 'min_cut_hmetis'})
        both_min_cuts['min'] = both_min_cuts.apply(lambda row: (row['min_cut_kahypar']
                                                               if row['min_cut_kahypar'] < row[
                                                               else row['min_cut_hmetis']), axi

        #calculate precentage of derivation of mean cuts from min cut for kahyper
        kahypar_percentages = pd.DataFrame(kahypar_data.groupby('graph')['cut'].mean()).reset_index()
        kahypar_percentages = pd.merge(kahypar_percentages, both_min_cuts[['graph','min']], on='graph')
        kahypar_percentages = kahypar_percentages.rename(columns={'cut' : 'mean_cut', 'min' : 'min_cut'
        kahypar_percentages['percent_deviation'] = kahypar_percentages.apply(lambda row : ((row['mean_c

        #calculate plot data for adaptive stopping rule
        kahypar_plot = pd.DataFrame({'deviation_leq' : np.arange(0,23)})
        kahypar_plot['num_graphs'] = kahypar_plot.apply(lambda row : (len(kahypar_percentages[kahypar_p
        kahypar_plot['percentage_of_graphs'] = kahypar_plot.apply(lambda row : (len(kahypar_percentages

        #calculate precentage of derivation of mean cuts from min cut for hmetis
        hmetis_percentages = pd.DataFrame(hmetis_data.groupby('graph')['cut'].mean()).reset_index()
        hmetis_percentages = pd.merge(hmetis_percentages, both_min_cuts[['graph','min']], on='graph')
        hmetis_percentages = hmetis_percentages.rename(columns={'cut' : 'mean_cut', 'min' : 'min_cut'})
        hmetis_percentages['percent_deviation'] = hmetis_percentages.apply(lambda row : ((row['mean_cut

        #calculate plot data for simple stopping rule
        hmetis_plot = pd.DataFrame({'deviation_leq' : np.arange(0,23)})
        hmetis_plot['num_graphs'] = hmetis_plot.apply(lambda row : (len(hmetis_percentages[hmetis_perce
        hmetis_plot[’percentage_of_graphs’] = hmetis_plot.apply(lambda row : (len(hmetis_percentages[hm

        # the plot
        #axis = kahypar_plot.plot(x=’deviation_leq’, y=’percentage_of_graphs’, label=’KaHyPar’, title=’
        #axis.set_ylabel(’Percentage of Graphs’)
        #hmetis_plot.plot(ax=axis, x=’deviation_leq’, y=’percentage_of_graphs’, label=’hmetis’, style=’
        #axis.set_xlabel(’Deviation from min-cut £[\leq \%]£’)
        #axis.get_xaxis().set_ticks(np.arange(25))
        #axis.get_yaxis().set_ticks(np.arange(0,105,5))
        #plt.gcf().set_size_inches(14.5,6.5)
        #plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

```
In [2]: from scipy import stats
        print ’hMetis: geometric mean (min_cuts)=’, stats.gmean(hmetis_min_cuts[’cut’])
        print ’KaHyPar: geometric mean (min_cuts)=’, stats.gmean(kahypar_min_cuts[’cut’])
        both_min_cuts
```

```
hMetis: geometric mean (min_cuts)= 404.070526294
KaHyPar: geometric mean (min_cuts)= 401.811406887
```

```
Out[2]:            graph  min_cut_kahypar  min_cut_hmetis   min
        0    avqlarge.hgr              143             142   142
        1    avqsmall.hgr              143             142   142
        2    bcsstk32.hgr             4667            4667  4667
        3    crystk01.hgr              420             420   420
        4         cs4.hgr              363             373   363
        5       ibm03.hgr              958             958   958
        6       ibm04.hgr              586             586   586
        7       ibm05.hgr             1724            1723  1723
        8   industry2.hgr              178             179   178
        9     memplus.hgr             5423            5691  5423
        10     s15850.hgr               56              57    56
        11     s35932.hgr               43              43    43
        12     s38584.hgr               49              49    49
        13    s3rmq4m1.hgr             360             360   360
        14    vibrobox.hgr            1990            1990  1990

        [15 rows x 4 columns]
```

```
In [3]: kahypar_avg_cuts = pd.DataFrame(kahypar_data.groupby(’graph’)[’cut’].mean()).reset_index()
        hmetis_avg_cuts = pd.DataFrame(hmetis_data.groupby(’graph’)[’cut’].mean()).reset_index()
        both_avg_cuts = pd.DataFrame(kahypar_avg_cuts)
        both_avg_cuts = both_avg_cuts.rename(columns={’cut’ : ’avg_cut_kahypar’})
        both_avg_cuts = pd.merge(both_avg_cuts, hmetis_avg_cuts, on=’graph’)
        both_avg_cuts = both_avg_cuts.rename(columns={’cut’ : ’avg_cut_hmetis’})
        print ’hMetis: geometric mean (avg_cuts)=’, stats.gmean(both_avg_cuts[’avg_cut_hmetis’])
        print ’KaHyPar: geometric mean (avg_cuts)=’, stats.gmean(both_avg_cuts[’avg_cut_kahypar’])
        both_avg_cuts
```

```
hMetis: geometric mean (avg_cuts)= 408.408886302
KaHyPar: geometric mean (avg_cuts)= 424.488351247
```

```
Out[3]:            graph  avg_cut_kahypar  avg_cut_hmetis
        0    avqlarge.hgr           158.96           142.8
        1    avqsmall.hgr           156.92           143.1
```

```
2    bcsstk32.hgr          4767.87          4756.7
3    crystk01.hgr           420.00           420.0
4         cs4.hgr           370.81           380.4
5       ibm03.hgr           971.04           961.6
6       ibm04.hgr           619.74           591.2
7       ibm05.hgr          1735.19          1727.5
8   industry2.hgr           202.06           185.1
9     memplus.hgr          5514.20          5754.7
10     s15850.hgr            60.69            57.8
11     s35932.hgr            43.00            43.0
12     s38584.hgr            52.14            49.0
13   s3rmq4m1.hgr           360.00           372.6
14   vibrobox.hgr          2480.16          1990.0

[15 rows x 3 columns]
```