Column Generation Techniques for GAP

July 22, 2019

1 Preliminary Tests

Github page : `https://github.com/mody3062/CG`

Testing algorithms

- Classical column generation (Kelly's cutting plane)
- Stabilized column generation (O. Du Merle, et. al., 1997)
- Separation + Classical column generation
- Separation + Stabilized column generation

Algorithmic parameters RMP was constructed with a single decision variable which is dummy. The coefficient of the dummy variable on the objective function was set to a sufficiently large value, which is the sum of listed values such that `np.sum(c,axis=1)`. For stabilized column generation algorithm, I changed the parameter ϵ from 0.01 to 0.0001 for every 100 trials. (I am not sure whether I could understand the criteria for changing the parameter value(ϵ) well.)

To start the column generation procedure, an initial re- stricted master problem has to be provided. This initial restricted master problem must have a feasible LP relax- ation to ensure that proper dual information is passed to the pricing problem. We have chosen to start with one column for each agent, corresponding to the optimal knap- sack solution, and a dummy column consisting of all ones with a large negative profit. The dummy column ensures that a feasible solution to the LP relaxation exists. This dummy column will be kept at all nodes of the branch-and- bound tree for the same reason.

	Kelly			Stab.			Sep.			Sep.+Stab.		
	iteration	total(s)	M(%)	iteration	total(s)	M(%)	iteration	total(s)	M(%)	iteration	total(s)	M(%)
d05100	2485	26.13	25%	2216	33.27	32%	2296	44.57	61%	2321	49.8	66%
d10100	1223	7.83	25%	1068	11.49	42%	949	6.38	32%	858	7.25	40%
d10200	4485	133.04	54%	4423	199.37	61%	3253	132.49	67%	3703	193.21	73%
d20100	852	3.67	26%	782	6.63	49%	673	3.24	29%	629	4.15	41%
d20200	2513	38.59	41%	2549	65.75	55%	1862	32.86	51%	2288	52.38	59%
e05100	2577	15.03	42%	2356	27.73	51%	3487	74.37	86%	2761	37.61	76%
e10100	1345	6.83	36%	1405	12.45	52%	1160	6.64	54%	1261	8.93	58%
e10200	6273	153.91	77%	6580	271.2	80%	4328	186.97	87%	4455	218.22	88%
e20100	942	3.13	32%	979	7.4	52%	711	2.81	36%	644	3.5	46%
e20200	3273	37.77	57%	3374	76.73	69%	2390	44.17	75%	2713	61.78	78%

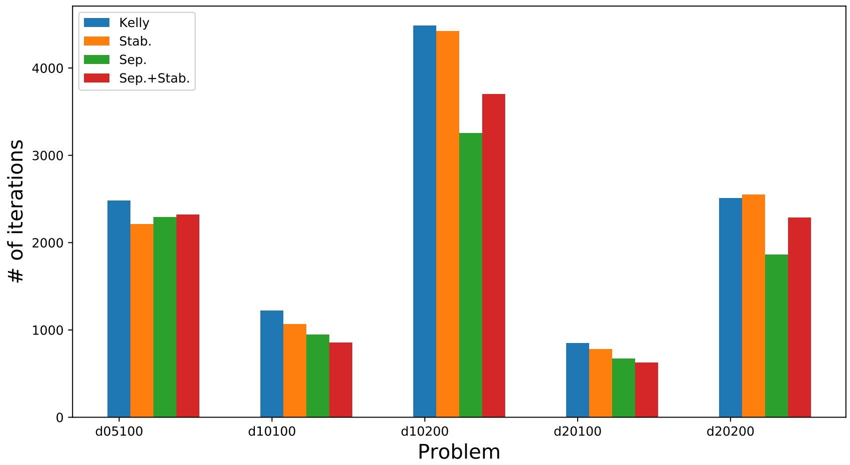

Figure 1: Performance comparison of the algorithms (gap_d)

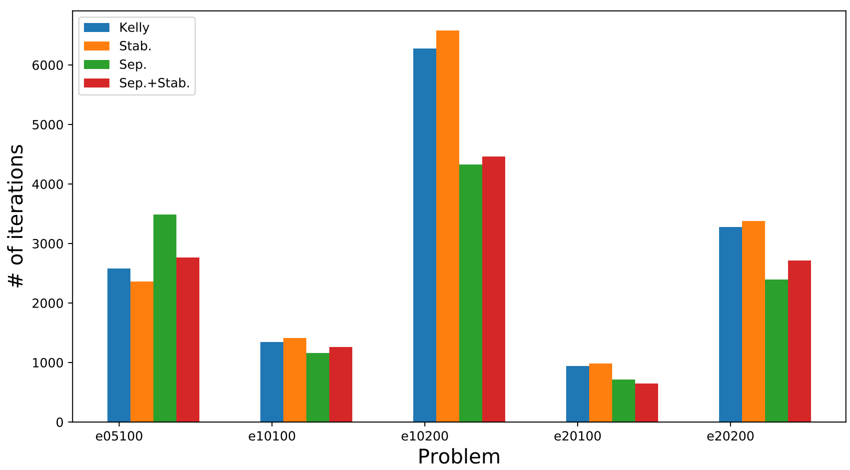

Figure 2: Performance comparison of the algorithms (gap_e)