An experimental evaluation of incremental and hierarchical *k*-median algorithms

David P. Williamson

(Cornell University, on sabbatical at TU Berlin)

Joint work with

Chandrashekhar Nagarajan

(Yahoo!)

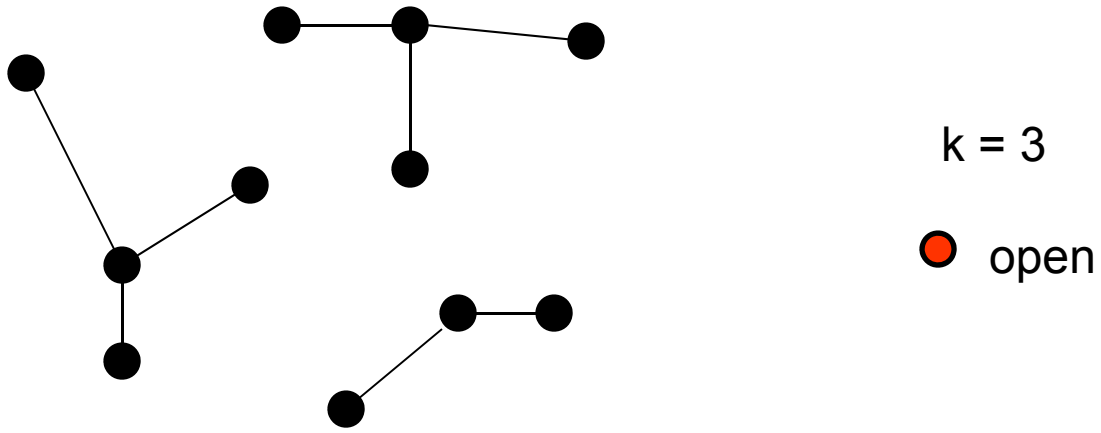
10th International Symposium on Experimental Algorithms

SEA 2011

May 6, 2011

The *k*-median problem

- *Input:* A set of points (*P*) in a metric space and an integer *k*. Let |*P*| = n, c_{ij} be the distance from *i* to *j.*
- *Output:* A set *S* of *k* points to "open" minimizing the sum of distances of each point to the nearest open point.



The k-median problem

- NP-hard, so work done on approximation algorithms
- An α-approximation algorithm is a polynomial time algorithm for finding a solution S such that

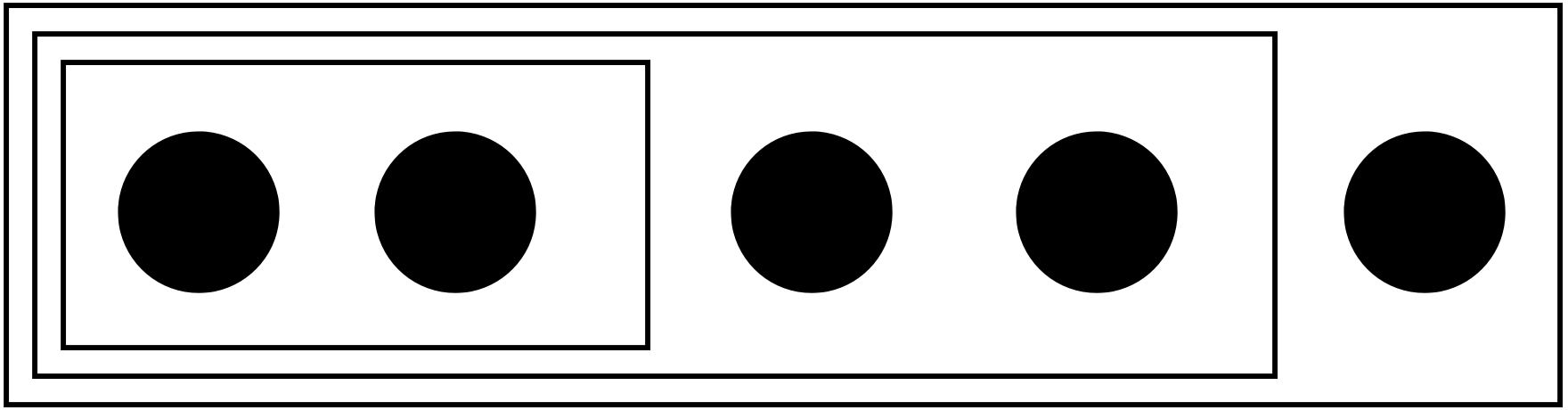
$$cost(S) = \sum_{j \in D} \min_{i \in S} c_{ij} \leq \alpha \; OPT_k$$

where OPT_k is cost of optimal solution on k points.

	Factor	Method
Charikar, Guha, Shmoys, Tardos (1999)	6.66	LP rounding
Jain, Vazirani (1999)	6	Primal-dual algorithm with Lagrangean relaxation
Jain, Mahdian, Markakis, Saberi, Vazirani (2003)	4	Greedy algorithm with Lagrangean relaxation
Arya, Garg, Khandekar, Meyerson, Munagala, Pandit (2001)	3 + ε	Local search

The incremental k-median problem

- Goal: find a sequence of all the points such that opening the first *k* in the sequence is a near-optimal solution to the *k*-median problem, for every *k*.



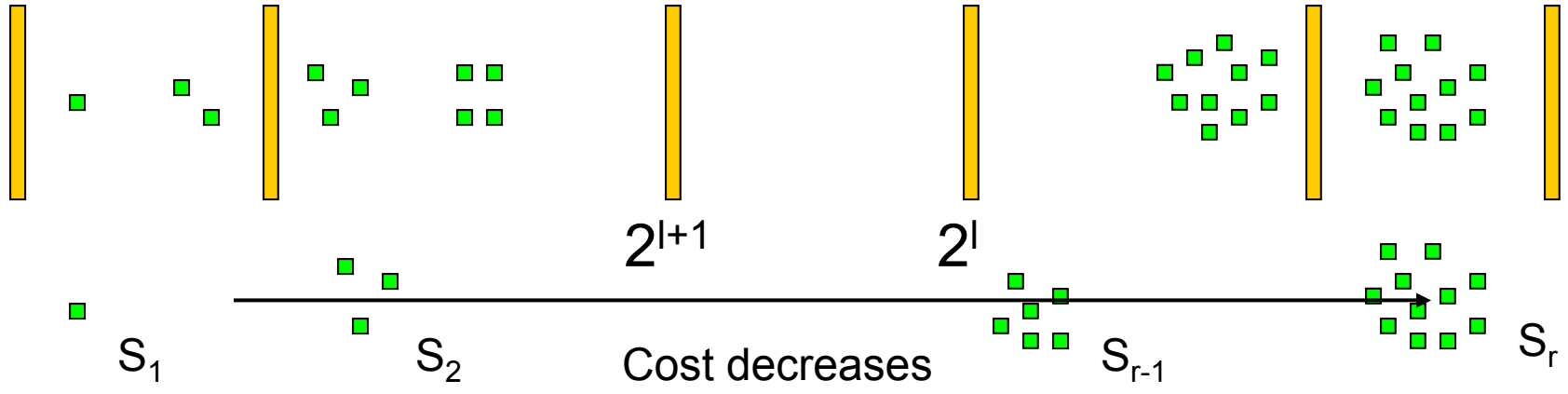
The incremental *k*-median problem

- *Input:* Same as the k-median problem without the integer *k*
- *Output:* An ordering of points $i_1, i_2, \ldots, i_n$
- *Competitive ratio* = $\max_k \frac{cost(\{i_1, i_2, \cdots, i_k\})}{OPT_k}$
- *Goal:* Find an ordering with minimum competitive ratio.

	Competitive ratio	Note
Mettu and Plaxton (2003)	29.86	Complicated greedy algorithm
Chrobak, Kenyon, Noga, Young (2008) Lin, Nagarajan, Rajaraman, W (2010)	24 + ε	Uses *k*-median α-approx alg as a black box (factor is 8α)
Lin, Nagarajan, Rajaraman, W (2010)	16	Uses Langrangean multiplier preserving facility location algorithm

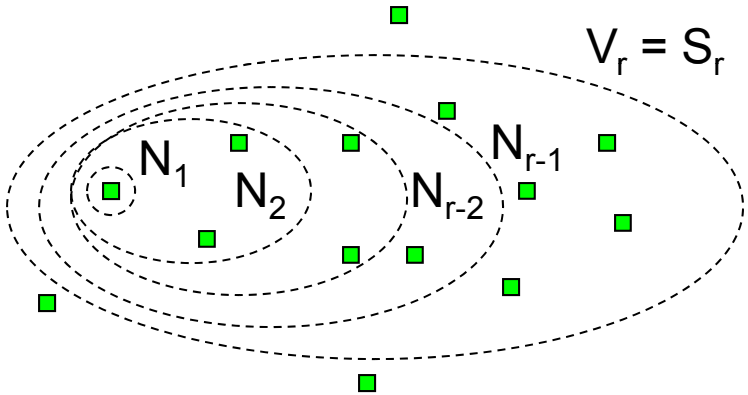
Incremental *k*-median algorithm

- Use an α-approximation algorithm for the *k*-median problem to get solutions for $k = 1$ to n.
- Bucket these solutions according to their costs into buckets of form $(2^l, 2^{l+1}]$ for integer l.
- Pick the maximum cost solutions from each bucket to obtain $S_1, S_2, \ldots, S_r$.

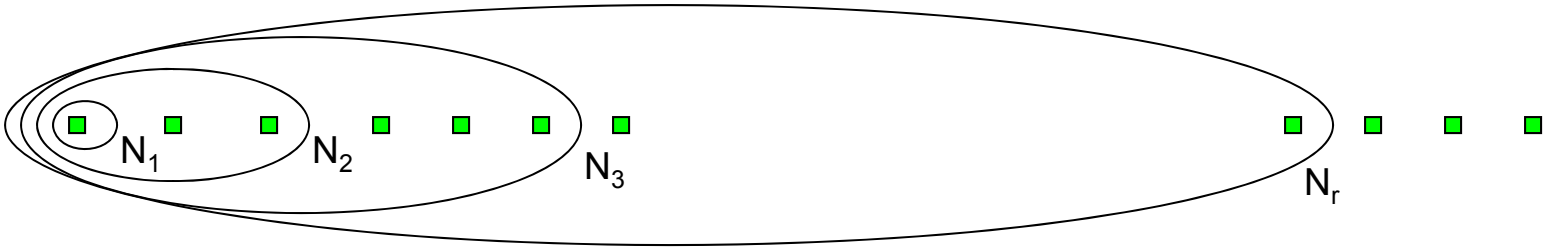


Algorithm (cont'd)

- Recursively combine these solutions using a nesting routine

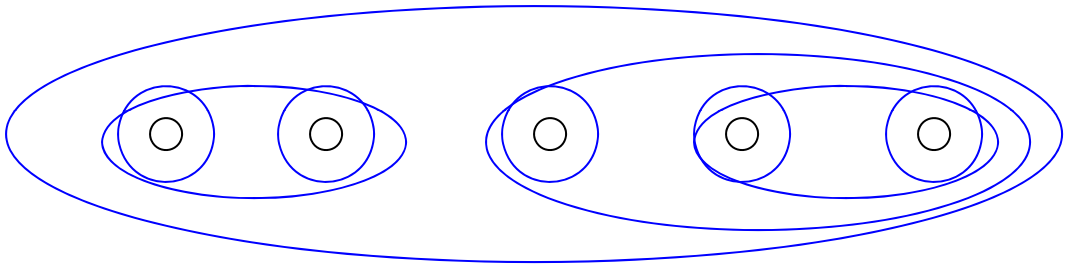


- Obtain an ordering maintaining this nesting



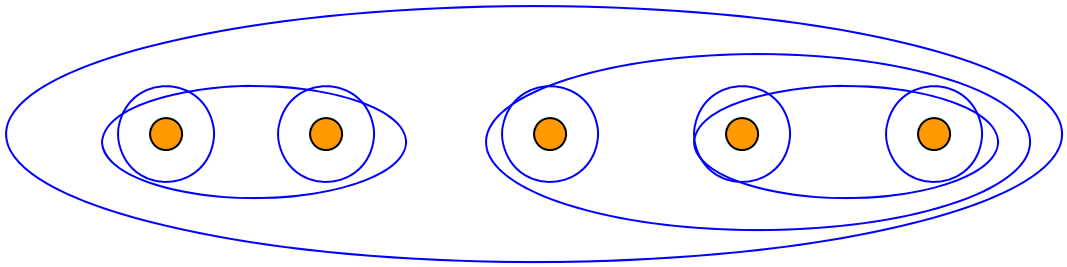
Hierarchical clustering

- Hierarchical clustering:
  - Collection of *k*-clusterings for all values of *k*
  - (k-1)-clustering is formed by merging two clusters in k-clustering.



Hierarchical clustering with cluster centers

- Hierarchical clustering with a center for each cluster in the clustering
- Merged clusters center should be one of the two original centers



Hierarchical median problem

- *Output:* Hierarchical clustering with cluster centers.
- *Clustering cost* = sum of the distances of points to their cluster center.
- *Competitive ratio* = $\max_k \frac{k\text{-}clustering\ \ cost}{OPT_k}$
- *Objective:* Minimize the competitive ratio.

	Competitive ratio	Note
Plaxton (2006)	238.9	Uses incremental *k*-median β-competitive alg as a black box (factor is 8β)
Lin, Nagarajan, Rajaraman, W (2010)	62.13	Uses *k*-median α-approx alg as a black box (factor is 20.71α)
Lin, Nagarajan, Rajaraman, W (2010)	48	Uses Langrangean multiplier preserving facility location algorithm

Experimental Results

- Comparisons
  - *k*-median algorithms
  - Incremental *k*-median algorithms
  - Hierarchical *k*-median algorithms

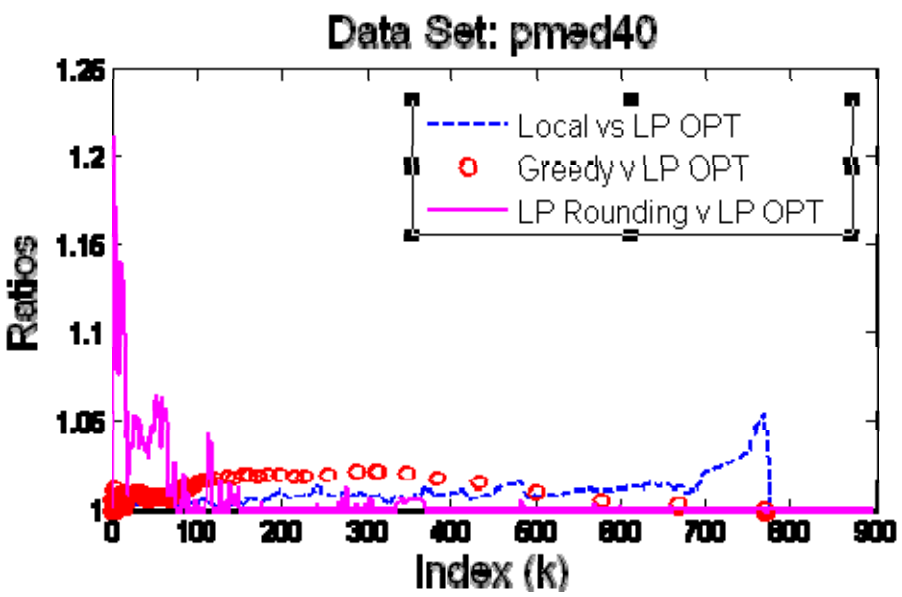
- The *k*-median datasets
  - *OR* library datasets (40) [Beasley 1985]
  - Galvao datasets (2) [Galvao and ReVelle 1996]
  - Alberta dataset [Alp, Erkut, Drezner 2003]

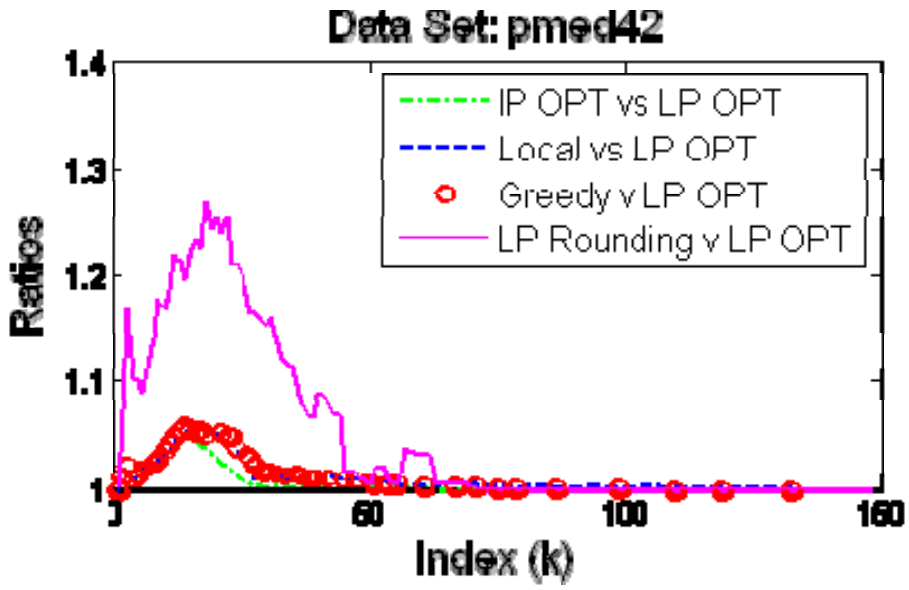
- We compare the quality of solutions and the running times of these algorithms on the datasets

The *k*-median algorithms

- We compare the following k-median algorithms on the datasets
  - CPLEX LP optimum
  - CPLEX IP optimum
  - Single swap local search algorithm (α=5)
    - Arya, Garg, Khandekar, Meyerson, Munagala, Pandit [2004]
  - Greedy facility location algorithm (α=2)
    - Jain, Mahdian, Markakis, Saberi, Vazirani [2003]
  - LP rounding algorithm (α=8)
    - Charikar, Guha, Tardos, Shmoys [1999]

Quality of the k-median solutions





Quality of k-median solutions

Dataset	n	IP OPT/LP OPT		Local /LP OPT		LPR/LP OPT	
		Mean	Max	Mean	Max	Mean	Max
pmed5	100	1	1.002	1.0042	1.0177	1.0026	1.1311
pmed10	200	1.0001	1.0097	1.0046	1.027	1.0008	1.0688
pmed15	300	1	1.0024	1.007	1.0476	1.0011	1.0472
pmed20	400	1.0001	1.0121	1.0073	1.0299	1.0022	1.1071
pmed25	500	1.0001	1.0035	1.0062	1.0218	1.0024	1.1034
pmed30	600	0	0	1.0068	1.0189	1.0036	1.1529
pmed34	700	0	0	1.006	1.0205	1.0061	1.2068
pmed37	800	0	0	1.0089	1.0323	1.0059	1.3148
pmed40	900	0	0	1.009	1.0543	1.0053	1.2123
Galvão100	100	1.0029	1.045	1.0041	1.0459	1.0352	1.2615
Galvão150	150	1.0048	1.0512	1.0096	1.0642	1.0479	1.2698
Alberta	316	1.0002	1.002	1.0132	1.0299	1.0026	1.0866

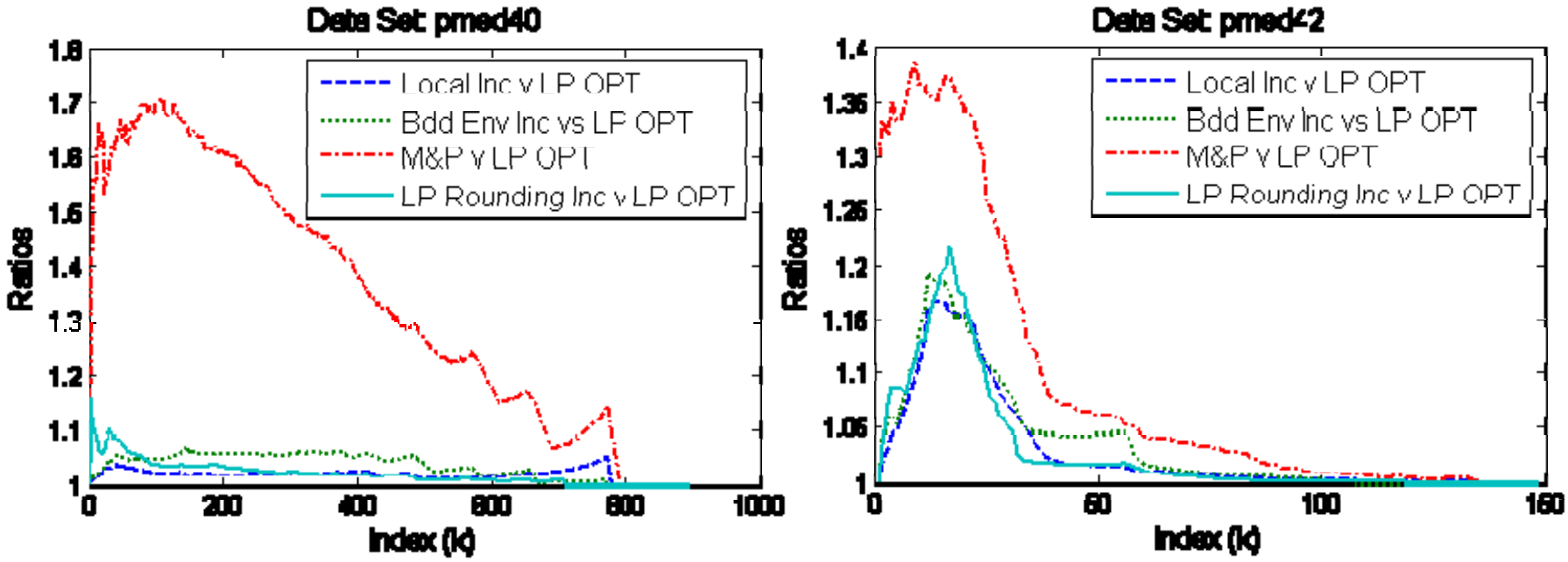
Running times of k-median algorithms

Dataset	n	Time in seconds to finish for all n				
		LP	IP	Local	Greedy	LPR
pmed5	100	11.484	39.078	34.656	170.03	11.641
pmed10	200	66.859	531.39	316.89	818.47	67.75
pmed15	300	204.39	2991.9	1302.8	2001.9	207.63
pmed20	400	669.33	11851	3620	4186.6	677.13
pmed25	500	1223.1	29149	8372.3	6419.6	1237.8
pmed30	600	4141.1	*	16890	9997.3	4166
Galvão100	100	12.641	1208.2	35.219	90.797	13.25
Galvão150	150	40.453	48039	129.7	256.13	41.875
Alberta	316	306.59	5248.5	1642.8	3791.1	311.11

Incremental *k*-median algorithms

- We compare the following incremental *k*-median algorithms
  - Our incremental *k*-median algorithm
    - using Arya et al.'s local search *k*-median solutions
      - 5-approximation algorithm → Competitive ratio = 40
    - using Jain et al.'s greedy facility location algorithm
      - 2-approximation algorithm → Competitive ratio = 16
    - using Charikar et al.'s LP rounding solutions
      - 8-approximation algorithm → Competitive ration = 64
  - Mettu and Plaxton's incremental *k*-median algorithm (Competitive Ratio = 29.86)

Quality of incremental k-median solutions



Quality of incremental k-median solutions

		LInc/LP OPT		GInc/LP OPT		MPInc/LP OPT		LPRInc/LP OPT	
Dataset	n	Mean	Max	Mean	Max	Mean	Max	Mean	Max
pmed5	100	1.0078	1.03	1.0619	1.1593	1.1812	1.4791	1.0127	1.0938
pmed10	200	1.0276	1.0562	1.0521	1.1043	1.3347	1.6957	1.0168	1.0467
pmed15	300	1.0214	1.0544	1.0404	1.0721	1.3826	1.7325	1.018	1.0306
pmed20	400	1.0273	1.0489	1.0385	1.0757	1.2429	1.4529	1.0256	1.0691
pmed25	500	1.0204	1.0395	1.0399	1.071	1.3522	1.7349	1.0215	1.0478
pmed30	600	1.0179	1.0466	1.0362	1.0722	1.3233	1.8021	1.0192	1.0639
pmed34	700	1.0202	1.049	1.0374	1.07	1.3768	1.7976	1.0172	1.0836
pmed37	800	1.0204	1.041	1.0356	1.0709	1.3405	1.7722	1.0189	1.0853
pmed40	900	1.0194	1.0543	1.035	1.0689	1.3397	1.7058	1.0208	1.161
Galvão100	100	1.0209	1.1601	1.024	1.1601	1.0711	1.3496	1.036	1.2205
Galvão150	150	1.0283	1.1655	1.0349	1.1925	1.0908	1.388	1.0285	1.2168
Alberta	316	1.0409	1.1967	1.0499	1.1432	1.157	1.4322	1.027	1.1223

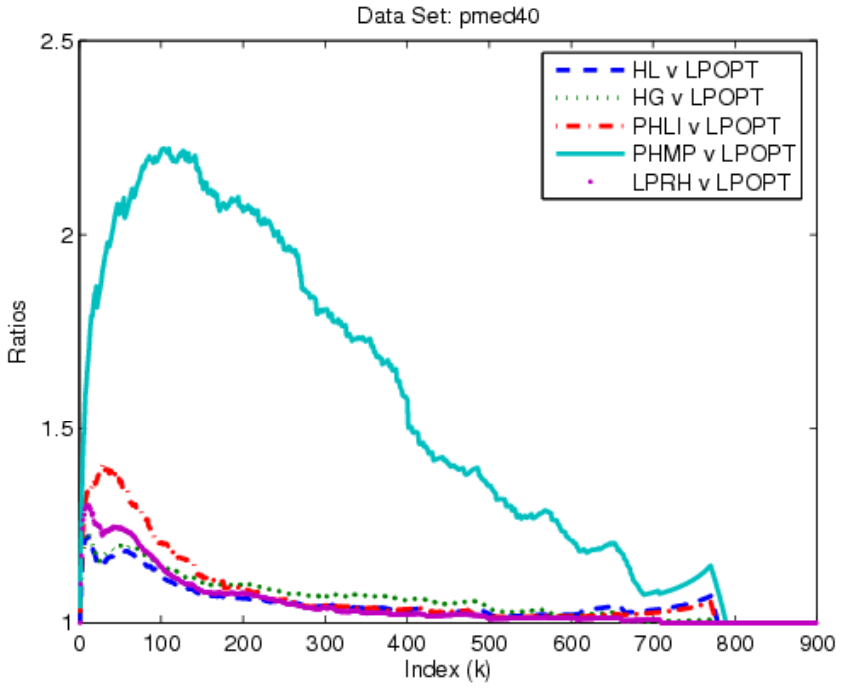
Running times of Incremental *k*-median algorithms

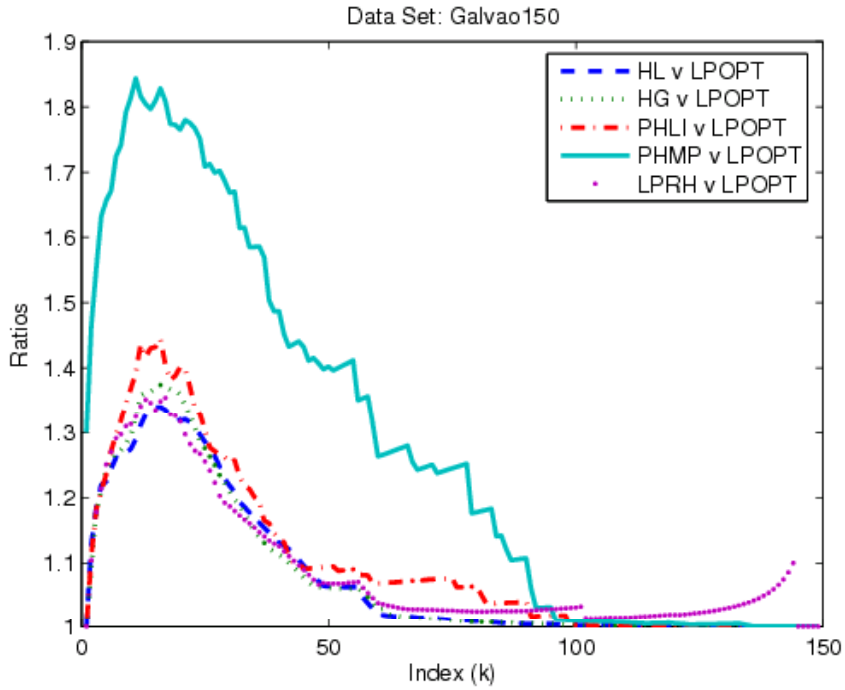
Dataset	Size	Running time in secs			
		IncL	IncG	MP	IncLPR
pmed5	100	34.766	170.17	0.578	11.797
pmed10	200	317.8	819.36	2.375	68.672
pmed15	300	1306.1	2005.2	5.797	210.95
pmed20	400	3628.8	4194.8	11.031	685.36
pmed25	500	8390.4	6437.8	18.813	1255.2
pmed30	600	16921	10029	30.297	4196.9
pmed34	700	30655	15461	43.391	*
pmed37	800	52134	33525	61.735	*
pmed40	900	*	*	81.673	*
Galvão100	100	35.36	90.937	0.563	13.422
Galvão150	150	130.05	256.45	1.281	42.265
Alberta	316	1646.6	3794.9	6.375	315.03

Hierarchical *k*-median algorithms

- We compare the following hierarchical *k*-median algorithms
  - Our hierarchical *k*-median algorithm
    - using Arya et al.'s local search *k*-median solutions
    - using Jain et al.'s bounded envelope
    - using Charikar et al.'s LP rounding solutions
  - Plaxton's hierarchical *k*-median algorithm
    - using Mettu and Plaxton's incremental *k*-median solutions
    - using our incremental *k*-median solutions obtained by using Arya et al.'s *k*-median solutions

Quality of hierarchical k-median solutions





Quality of hierarchical k-median solutions

		HL/LP		HG/LP		PHLI/LP		PHMP/LP		LPRH/LP	
Dataset	n	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
pmed5	100	1.017	1.125	1.071	1.159	1.015	1.111	1.268	1.782	1.021	1.18
pmed10	200	1.044	1.14	1.068	1.126	1.049	1.17	1.449	1.992	1.031	1.151
pmed15	300	1.041	1.154	1.062	1.186	1.051	1.19	1.579	2.221	1.039	1.163
pmed20	400	1.051	1.172	1.067	1.197	1.07	1.301	1.349	1.739	1.047	1.174
pmed25	500	1.041	1.145	1.061	1.183	1.059	1.256	1.535	2.107	1.04	1.151
pmed30	600	1.039	1.19	1.059	1.183	1.056	1.319	1.539	2.301	1.04	1.247
pmed34	700	1.046	1.194	1.063	1.183	1.066	1.308	1.584	2.234	1.045	1.285
pmed37	800	1.049	1.241	1.06	1.212	1.072	1.42	1.519	2.261	1.047	1.244
pmed40	900	1.05	1.222	1.06	1.23	1.071	1.405	1.528	2.223	1.048	1.302
Galvão100	100	1.05	1.274	1.05	1.36	1.068	1.373	1.236	1.811	1.092	1.433
Galvão150	150	1.078	1.34	1.078	1.374	1.103	1.445	1.282	1.844	1.091	1.354
Alberta	316	1.064	1.266	1.074	1.248	1.082	1.283	1.267	1.716	1.046	1.218

Summary of experimental results

- Charikar et al.'s LP rounding *k*-median algorithm works faster
- Arya et al.'s local search *k*-median algorithm gives better solutions
- Our incremental algorithms better in terms of quality than Mettu and Plaxton's algorithm
- Our hierarchical algorithms are better in terms of quality than Plaxton's algorithm, even when Plaxton is given our incremental solutions
- However the running times of our incremental algorithms are much worse than Mettu and Plaxton's algorithm
- Main takeaway: despite the strong constraints of coming up with incremental and hierarchical solutions, these algorithms deliver quite good solutions in practice.

Open questions

- Implementations are slowed because we need calculate approximate k-medians for all values of k, but we really only need the solution of at most cost 2^l for all values of l.
  - Can this be computed faster than by doing binary search?
  - Or maybe another primitive could find a solution per bucket?
- How do these algorithms compare with standard hierarchical clustering algorithms that have no provable competitive ratio?

Thanks. Any questions?