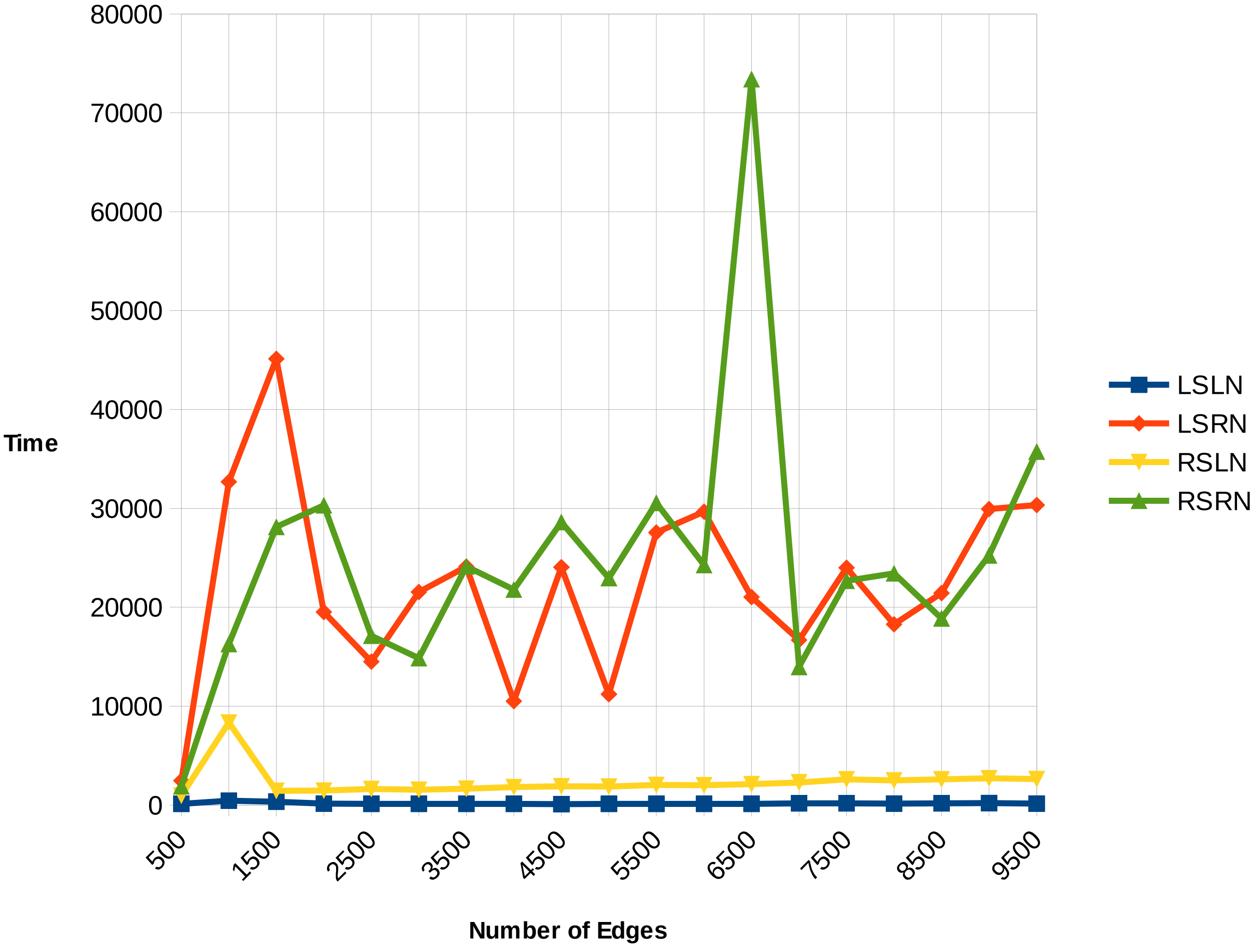
1. Remote searcher, local nodes: call of getDistance method on remote searcher requires nodes (the entire graph sometimes) to be transfered over network. Because NodeImpl is not remotely accessible it is serialized. During the serialization all neighbors of given nodes are processed recursively.
2. Local searcher, remote nodes: local searcher accesses the nodes by remote reference. This remote reference does not contain any data. So when `getNeighbors` is called neighbor nodes are transfered from the server.
3. Remote searcher, remote nodes: remote searcher assesses the nodes by remote reference. When getNeighbors is called, request is sent to the client and from client back to the server which returns a set of nodes. This way is traversed back to client and finally server.

We have used random graph with 500 vertices and variable number of edges. Same graph for all mentioned methods has been generated. Following figure shows dependency of number of edges in a random graph and time of `getDistance()` call for randomly chosen nodes (for each methods the same nodes). For each number of edges calculation has been performed 10 times and we took the average.

Legend:



LSLN - Local searcher, local nodes
LSRN - Local searcher, remote nodes
RSLN - Remote searcher, local nodes
RSRN - Remote searcher, remote nodes

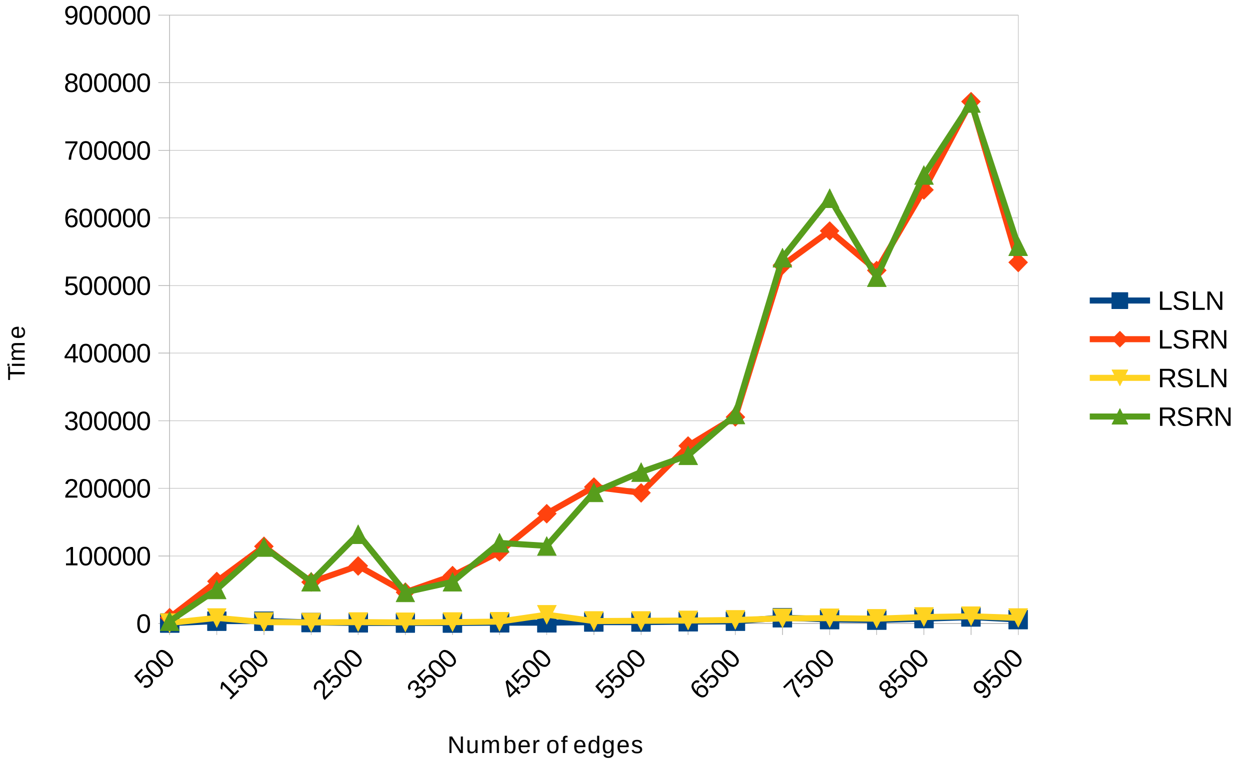
Observation: Totally local search (LSLN) is the most efficient as nothing is transfered over network and nodes are accessible directly. Almost the same case is the RSLN method where the graph is transfered over network and the searching is performed locally (but on the server). This method would be useful if server is a high performance computer while client is just regular personal computer. The performance of the last two methods (LSRN and RSRN) is decreased by the fact that every step to the next node in the backtracking algorithm is a call over network.

Measured data:

Average - Time	Method			
Edges	**LSLN**	**LSRN**	**RSLN**	**RSRN**
500	129,7	2461,7	957,7	1892,3
1000	439,7	32689,8	8355,5	16284
1500	346,8	45130,1	1446,8	28131,8
2000	138,1	19510,9	1458,3	30305,7
2500	121,1	14492,6	1638,1	17111,8
3000	115	21552,5	1540,8	14855
3500	120,2	24118,7	1665,1	24136,7
4000	125,3	10495,3	1818,9	21791,1
4500	81,8	24064,6	1905,6	28623,2
5000	113,3	11221	1870	22946
5500	119,8	27568,6	2020,3	30560,2
6000	109,6	29659,3	2002,5	24271,2
6500	120,6	21031,8	2112,2	73404,5
7000	181,6	16698,4	2290	13975,8
7500	170,8	23995,2	2606,7	22679,2
8000	144,4	18276	2492,9	23444,9
8500	182,4	21431,3	2596	18886,9
9000	202,3	29944,5	2718,3	25231,2
9500	153,6	30347,1	2616,6	35732,6

We have also measured times of `getTransitiveDistance()` with parameters 2, 3 and 4. Times of these operations seems to fall down when the graph becomes dense enough. As in the previous measurement we have repeated the experiment for each graph 10 times and made an average.

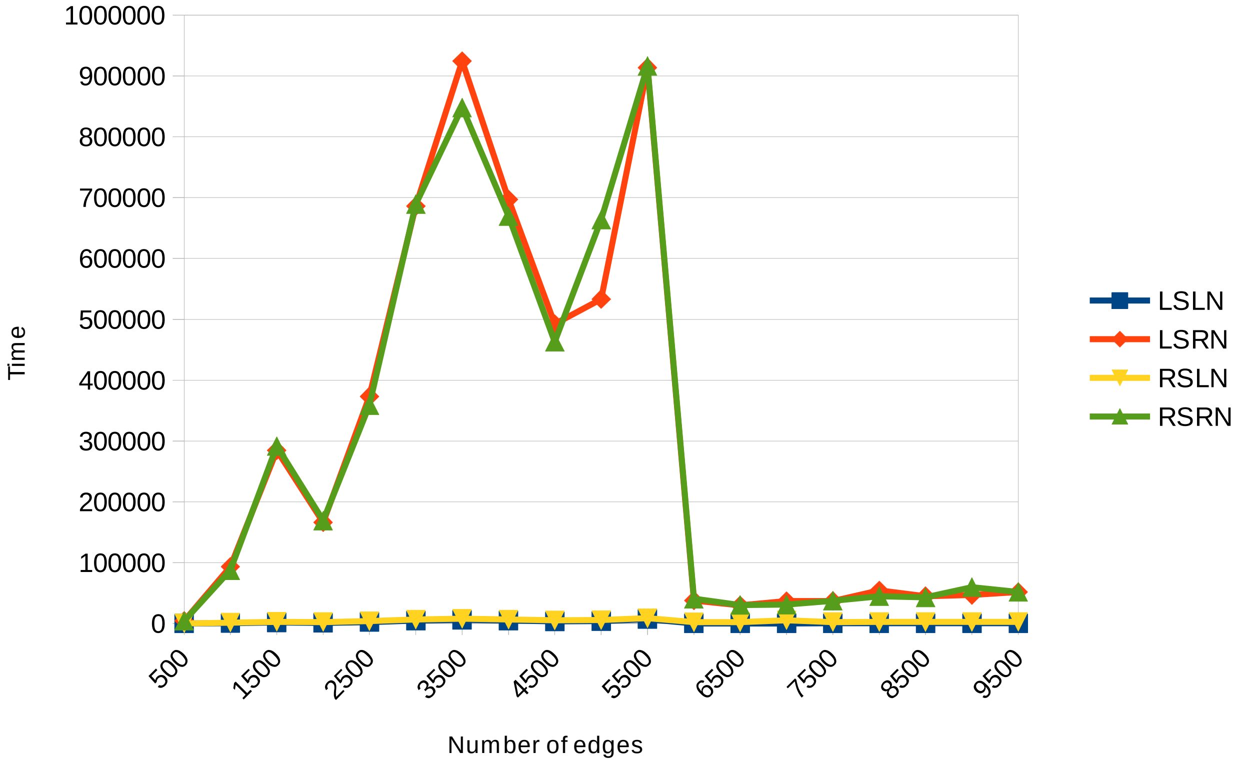
Distance of 2



Tdistance 2

Average - TTime	Method			
Edges	LSLN	LSRN	RSLN	RSRN
500	537,5	8548,5	631,9	3304,8
1000	3571,5	62199,1	8231,5	50492,5
1500	3356,8	114051,5	2052,2	112821
2000	1364,8	61342	1671,4	61767,2
2500	976,3	85175,4	2049,6	131942,8
3000	604,8	46230,5	1784,9	45987,7
3500	758,7	70708,6	2142,8	61524,2
4000	1102,5	105886,6	2644,2	119273,7
4500	1072,6	162389,4	12875,9	114687,5
5000	2105,5	201969,2	3706,4	194024,4
5500	2105,3	193086,6	3826,8	223958
6000	2735,1	262797,8	4428,8	248533,1
6500	3281,3	305507,3	5241,6	308923,3
7000	8616,9	529932,3	7633,7	540881,2
7500	5767,7	580733,6	7838,8	629093,1
8000	4994,1	522172,4	7088,4	511999,5
8500	7248,8	641420,3	9507,4	663253,8
9000	9602,3	771929,7	10836	769920,1
9500	5831	534007,9	8312,7	557996,1

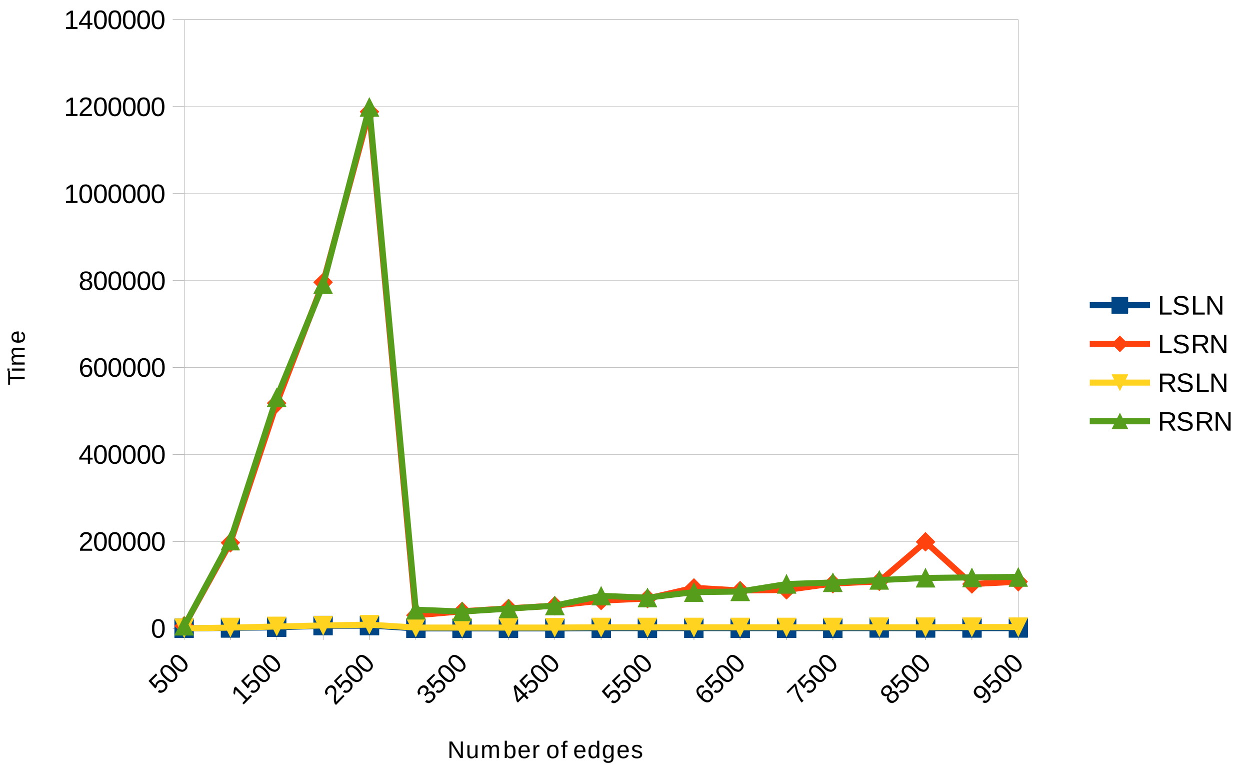
Distance of 3



Tdistance	3

Average - TTime	Method			
Edges	LSLN	LSRN	RSLN	RSRN
500	27,4	4431,8	135,8	4446,6
1000	478,1	93402,4	1484,1	87661,2
1500	1600,5	284654,8	2845,9	291504,3
2000	1058,1	166212,3	2334	169107,8
2500	2392,5	373196	3881,6	358748,5
3000	4697,4	685868,8	6459,2	689148,7
3500	5965,3	924296,9	7777,3	848173,7
4000	4698,9	696998	6573,7	669346,3
4500	3382,5	492132,3	5168,3	462870,4
5000	3750	533208,4	5608,8	663617,6
5500	7115,1	913602,8	8674,1	916361,8
6000	205,3	37885	1997	40702,5
6500	223	29813,2	2096,3	30347,5
7000	230,9	36891,8	5315,7	30969,1
7500	259,2	36955,7	2273,7	37415
8000	298,4	54281,7	2507,1	44850,7
8500	306	44833	2494,4	42950,9
9000	319,1	46756,4	2570,3	59638,2
9500	338,1	51733,1	2680,9	51824

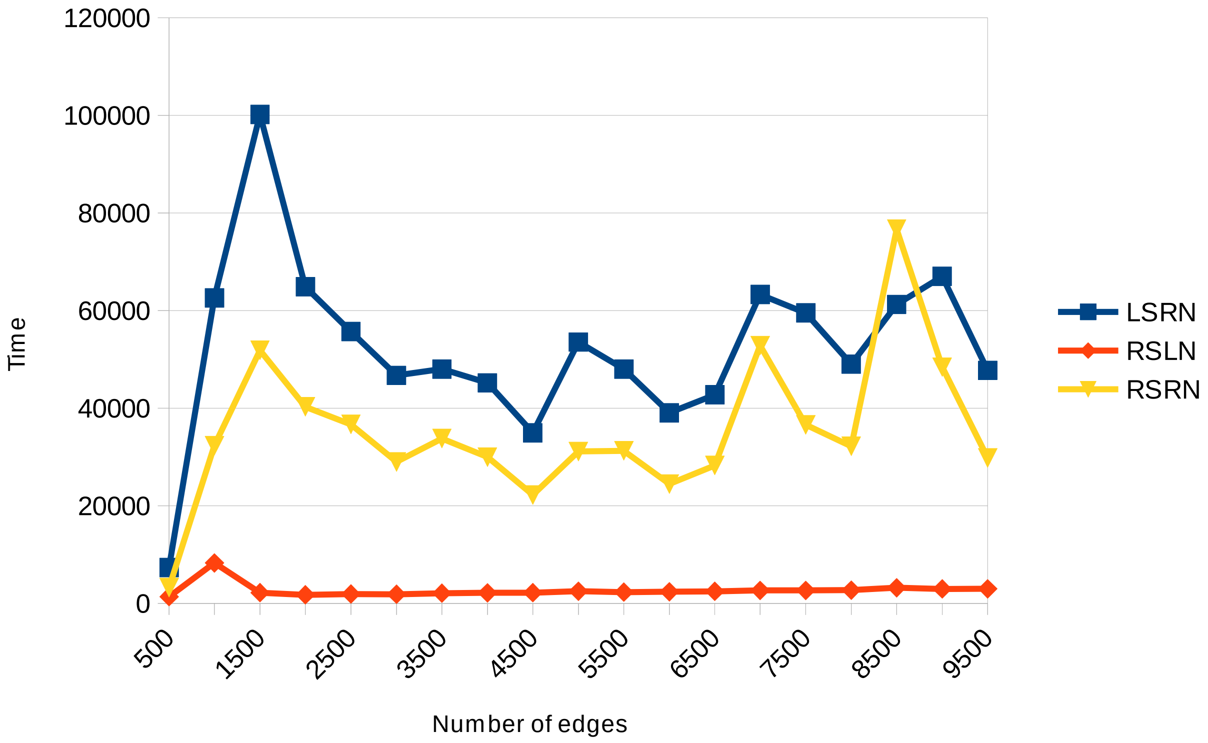
Distance of 4



Tdistance 4

Average - TTime	Method			
Edges	LSLN	LSRN	RSLN	RSRN
500	27,3	5665,2	131,2	5635,3
1000	922	196984,2	2000,4	201671,7
1500	2752,8	517880,5	4522,7	531357,7
2000	6418,9	795983,4	6544,9	791000,8
2500	6587,1	1188120,2	8540,2	1198743
3000	195,3	30287,2	1753	42796,9
3500	239,2	39166,1	1663,9	38886,2
4000	261,9	45887,5	1796,2	45418,7
4500	294,1	51929,5	1896,5	52493,2
5000	347,9	63955,4	2230,9	74862,1
5500	382,2	68608,8	2157,5	70721,3
6000	439,7	93351,6	2253,8	83415
6500	476	87106,7	2364,8	85027,8
7000	482,6	88378,8	2500	102182,7
7500	517,4	103014,6	2552,7	105755,2
8000	528,2	108059,4	2650,3	111140,4
8500	562	199067,5	2744,8	116284,2
9000	580,8	102035,8	2830,1	117023,6
9500	600,6	107206,9	2923,3	118383,2

The last experiment consists of running server on a physically different machine. We have chosen two computers from MS lab. So the difference is not so significant. Also average of 10 measurement has been done. LSLN method has been left as there is no difference. The results are very similar to the first experiment but the times at larger approximately twice. Notice that we have been using the same random graphs.


Time
Number of edges
LSRN
RSLN
RSRN