Matemática Contemporânea, Vol. 44, 1–20

An Experimental Analysis of Exact Algorithms for the Maximum Clique Problem

Cleverson Sebastião dos Anjos
Alexandre Prusch Züge **Renato Carmo**

Abstract

We perform an experimental analysis of 10 exact algorithms for the Maximum Clique problem. The comparative results reported imply that, when solving a particular maximum clique problem, if the focus is on minimizing the number of branching steps, the $\chi + \mathsf{df}$ algorithm is the best option. However, if running time is the main concern, then the dyn algorithm is the best choice.

1 Introduction

A *graph* G is a pair $(V(G), E(G))$ where $V(G)$ is a finite set of *vertices* and $E(G)$ is a set of (unordered) pairs of vertices, called *edges*. Two vertices u and v are *neighbors* in G if $\{u, v\} \in E(G)$. The graph G is *complete* if any two vertices of G are neighbors. If $X \subseteq V(G)$, then $G[X]$ is the *subgraph of G induced by X* given by $V(G[X]) = X$ and $E(G[X]) = \{\{x, y\} \in E(G) \colon x, y \in X\}$. The set X is a *clique* if

2000 AMS Subject Classification: 05C04.
Key Words and Phrases: maximum clique, algorithms, branch and bound.

the graph $G[X]$ is complete. The Maximum Clique Problem (MCP) is the ($\mathcal{NP}$–hard) problem of finding a clique of maximum size on a given graph.

There are a number of proposed algorithms for the exact solution of the MCP which are reported to effectively solve instances of practical interest (some of them of considerable size) in several domains. Among them, branch and bound based schemes stand out as the best approach in practice. In this work we report an experimental comparison of ten of these algorithms for the MCP, where we apply the concepts of *Experimental Algorithm Analysis* as proposed in [McG12].

One problem in comparing these algorithms is the way their merits are presented in the literature. Generally speaking, each author presents the outcome of their work by providing the results obtained by carrying out experiments concerning "their algorithm". The resulting comparison of experimental data, then, is achieved by comparing data from different implementations running under different computational environments.

Ten of these branch and bound algorithms for the MCP were described and implemented in [CZ12] under a unifying conceptual framework which leads naturally to an unified implementation of them as parameterized versions of a general branch and bound routine.

The text is organized as follows. In Section 2 we provide information regarding the computational problem, algorithms analyzed as well as the used instances. In Section 3 we describe the design model and environment (hardware and software) specifications. In Section 4 the experimental results are presented and discussed. Lastly, in Section 5 we present the conclusion. It should be noted that the experiments generated a large amount of data of which only an abridged version can be presented in this extended abstract. We refer the reader to [dA15] for a full discussion. Lastly, we present in the Appendix more extensive results.

2 Preliminaries

We compare 10 algorithms, namely, basic [CZ12], cp [CP90], df [Fah02], χ [Fah02], χ+df [Fah02], dyn [KJ07], mcr [TK07], mcq [TS03],mcs [TSH+10], and nobound, which is a variation of basic that uses no bounding function. Following [CZ12], we refer to all of them collectively as MCP BB algorithms (for "maximum clique problem branch and bound"). We note that the main difference amongst the MCP BB algorithms is the bounding function used. For more detailed information on all of the algorithms we refer the reader to [CZ12].

We used the graphs from the Second DIMACS Implementation Challenge [JT96], which are 66 graphs divided into 9 families. These families are referred to as c-fat [BP90], dsjc [JAMS91], mann (clique formulation of the Steiner Triple System Problem, translated from the set covering formulation), brock [BC96], gen [San92], hamming (graph on binary words with an edge if and only if the two words are at least at a certain Hamming distance apart), keller [LS92, CS90] and p_hat (graphs generated with the p-hat generator which is a generalization of the classical uniform random graph generator; graphs generated with p-hat have wider node degree spread and larger cliques than uniform graphs).

3 The Design Model and the Environment Specifications

The experimental analysis treats algorithms as laboratory subjects, emphasizing control of parameters, isolation of key components, model building, and statistical analysis. It combines elements from the empirical analysis, such as code and measurement and the abstract point of view of the theoretical approach [McG12].

We define the elements of the design model as proposed in [McG12]. This model may vary according to the intent of the experiment by registering different values such as concurrently running processes, different performance indicators and outputs.

We must define and register the **server** where the tests were executed; the **memory load**; the **question** to be answered by the experiment; the **performance indicator**, which is a dimension of the algorithm that can be measured; the **parameters**, which are a property of the algorithm that might affect the value of the performance indicator; the **levels**, which are the value attributed to each parameter during a test; the **number of tests** executed and the **output**.

We use the `latrappe` server. The memory load was registered at each run of a test and, for the sake of brevity, is not exposed here. The question is "How do the MCP BB algorithms behave with the DIMACS instances?". Our performance indicator is a combination of two indicators, CPU time and number of branching steps. The parameters are the number of vertices (n) and edges (m). Each DIMACS graph has its own sets of levels, which for sake of brevity are not mentioned here. The number of tests executed for each design point is one. Lastly, the outputs registered are maximum clique size, number of branching steps, CPU time and timeout.

The `latrappe` server has an Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz processor and 65 941 476 Kb of RAM memory. We used the Python 2.7.5+ [GCC 4.8.1] compiler without optimization. All of the algorithms were implemented by Züge [Zï1] using the SAGE platform [S+15].

A timeout limit of 38,600 seconds for each run was established. Lastly, we define our performance metric (a dimension of the algorithms performance that can be measured) by joining two performance indicators (an instance of the performance metric), namely the running time, which is a important performance indicator, but platform dependent, and the number of branching steps, which is platform independent.

4 Experimental Results and Conclusions

Given the limited amount of space we offer an overview of the results. A detailed description of the instances, algorithms and full results can be found in [dA15].

4.1 Size of the Maximum Clique Found

Table 1 displays the number of times each algorithm found the maximum clique (whether it completed execution or not). It is evident that dyn was the best at finding the Maximum Clique, followed closely by mcr.

dyn	mcr	mcs	basic	χ	$\chi + \mathsf{df}$	cp	mcq	df	nobound
52	50	43	32	32	32	32	32	30	26

Table 1: Number of maximum cliques found by each algorithm, ranked from best to worst.

4.2 The Number of Completed Executions

Algorithm dyn successfully identified the maximum clique in 45 of the 66 instances, which is a higher success rate than that achieved by any of the other algorithms. In Table 2 we can find the number of instances each algorithm completed without exceeding the time limit.

The full results can be found in Table 3, in the Appendix.

dyn	mcr	mcs	χ	$\chi + \mathsf{df}$	basic	df	mcq	cp	nobound
45	44	40	28	28	26	26	26	23	15

Table 2: Number of instances each algorithm was able to finish, ranked from best to worst.

4.3 The Number of Branching Steps

When it comes to the most successful algorithm in terms of the number of the branching steps, two algorithms stood out. The $\chi + \mathsf{df}$ algorithm was the best at 27 instances, followed by the χ algorithm, which was the best at 18 instances. Both algorithms use coloring in order to determine the bounding value. The algorithms calculate the chromatic number dynamically during their execution, which showed to be the best technique to reduce the number of branching steps amongst the ten algorithms we analyzed. The χ algorithm was also the only one that was the best regarding an entire family, the c-fat family. The full results can be found in Table 4 and Table 5, in the Appendix.

4.4 Running Time

It was evident that dyn was the best since it had the best running time in 39 of the 66 instances. Besides, it was the only algorithm that was able to solve the sanr200_0.9 instance. While the other algorithms halted at 38,600 seconds (timeout), dyn found the maximum clique in 24,012.09 seconds. Full results with respect to running time can be seen in Tables 6 and 7 in the Appendix.

In second place, the mcr algorithm showed the best performance in 6 instances. None of the 10 algorithms were able to solve the other 21 instances within the specified time frame. Still, in those cases dyn found the largest clique for 6 instances, tied with other algorithms in 10 cases and lesser in 4 cases to the algorithms mcr and mcs. The full results can be found in Table 3, in the Appendix.

4.5 Running Time and Number of Branching Steps

Lastly, we analyze the relationship between the running time and the number of branching steps.

We do so by dividing the overall running time t by the number of branching steps T. The ratio $\frac{t}{T}$ thus obtained can be found in Tables 8 and 9 in the Appendix.

The algorithms χ and $\chi + \mathsf{df}$ had the best results regarding the generation of branching steps, however, their performance was the worst among all 10 algorithms with respect to the measure when checking the relationship running time versus number of branching steps. The χ algorithm was the worst in 28 cases and $\chi + \mathsf{df}$ in the remaining 38.

These algorithms obtain the bounding value by computing colourings of the graph. We conclude that the bound estimated is of quality, but it demands much more time to be calculated.

The behavior that we observed in the 10 algorithms analyzed was that the more time was spent in estimating a bound, the fewer branching steps the algorithm generated. So it becomes clear that there is a tradeoff between these two measures, running time and branching steps.

5 Conclusion

It became clear that if the focus is on generating fewer branching steps (i.e. the algorithm that better applies the bounding technique), $\chi + \mathsf{df}$ is the best option, if the running time is the main concern, dyn is the best choice. Our results also indicated that the best bounding techniques demanded more time than the worse ones, indicating a tradeoff between the two performance indicators.

A Results from the Experiment

Here we present the results from the experiment carried out.

In Table 3 we can see the cases where an algorithm was able to complete its execution over an instance. An "X" denotes that it did so. Instances for which none of the algorithms were able to run to completion are not shown in the table.

The results regarding the number of branching steps were divided into two Tables 4 and 5 for better visualization.

Running time results are found also divided into Tables 6 and 7 .

Lastly, the results from the ratio $\frac{t}{T}$, where t is the running time and T is the number of branching steps, are presented in Tables 8 and 9.

	nobound	basic	χ	χ + df	cp	df	dyn	mcq	mcr	mcs
brock200_1		X			X		X	X	X	X
brock200_2	X	X	X	X	X	X	X	X	X	X
brock200_3	X	X	X	X	X	X	X	X	X	X
brock200_4		X	X	X	X	X	X	X	X	X
c-fat200-1	X	X	X	X	X	X	X	X	X	X
c-fat200-2	X	X	X	X	X	X	X	X	X	X
c-fat200-5		X	X	X		X	X	X	X	X
c-fat500-1	X	X	X	X	X	X	X	X	X	X
c-fat500-10			X	X		X	X		X	X
c-fat500-2		X	X	X	X	X	X	X	X	X
c-fat500-5		X	X	X	X	X	X	X	X	X
hamming10-2			X	X			X		X	X
hamming6-2		X	X	X	X	X	X	X	X	X
hamming6-4	X	X	X	X	X	X	X	X	X	X
hamming8-2			X	X			X		X	X
hamming8-4		X	X	X	X	X	X	X	X	X
johnson16-2-4	X	X			X	X	X	X	X	X
johnson8-2-4	X	X	X	X	X	X	X	X	X	X
johnson8-4-4	X	X	X	X	X	X	X	X	X	X
keller4	X	X	X	X	X	X	X	X	X	X
MANN_a27							X		X	
MANN_a9	X	X	X	X	X	X	X	X	X	X
p_hat1000-1	X	X			X	X	X	X	X	X
p_hat1500-1		X			X	X	X	X	X	X
p_hat300-1	X	X	X	X	X	X	X	X	X	X
p_hat300-2		X	X	X	X	X	X	X	X	X
p_hat300-3							X		X	
p_hat500-1	X	X	X	X	X	X	X	X	X	X
p_hat500-2							X		X	X
p_hat700-1	X	X	X	X	X	X	X	X	X	X
p_hat700-2							X		X	
san1000							X		X	X
san200_0,7_1			X	X			X		X	X
san200_0,7_2			X	X			X		X	X
san200_0,9_1			X	X			X		X	X
san200_0,9_2							X		X	X
san200_0,9_3							X		X	X
san400_0,5_1			X	X			X		X	X
san400_0,7_1							X		X	X
san400_0,7_2							X		X	X
san400_0,7_3							X		X	X
san400_0,9_1							X		X	
sanr200_0,7		X	X	X		X	X	X	X	X
sanr200_0,9							X			
sanr400_0,5		X			X	X	X	X	X	X

Table 3: Tests that had their execution concluded before the timeout.

Instance	nobound	basic	χ	χ + df	cp
brock200_1	792252455	876403943	491164	455516	258012511
brock200_2	12585199	700185	10069	9921	455413
brock200_3	365514611	8610083	74455	74663	3589195
brock200_4	653665494	28636755	132095	132071	15766703
brock400_1	1372832800	1579559329	143552	128869	1411505522
brock400_2	1373940291	1523953438	139388	139544	1403901415
brock400_3	1557768190	1544994800	133516	139726	1340757472
brock400_4	1368319292	1490862387	133002	137355	1406009465
brock800_1	1438947939	1702002635	97001	103395	1552537663
brock800_2	1409054091	1649041965	81990	96856	1583650878
brock800_3	1498118610	1620665373	91216	108616	1532947234
brock800_4	1494195878	1620236354	85968	114930	1489801507
c-fat200-1	162561	811	59	39	515
c-fat200-2	192856065	1917	49	5	2405
c-fat200-5	1364010710	140789	239	127	643874170
c-fat500-1	1125889	2151	29	5	1187
c-fat500-10	355706421	286282778	253	5	346751948
c-fat500-2	780134773	8365	53	5	4227
c-fat500-5	723569720	571779	129	5	29861813
hamming10-2	188347270	118146502	1025	543	111290446
hamming10-4	1862287451	1845277361	38020	39720	1672257761
hamming6-2	1525409292	508981	65	33	203817
hamming6-4	3937	2091	199	179	1657
hamming8-2	641221038	397723307	257	129	386744131
hamming8-4	1274021383	114637761	34299	28735	33923933
johnson16-2-4	92413471	36684707	1903631	1937023	23284309
johnson32-2-4	3429861301	3118368201	653465	777837	3078378672
johnson8-2-4	1527	717	81	79	569
johnson8-4-4	7705151	272001	141	131	87663
keller4	127267065	19055823	64357	60083	8518051
keller5	1435953623	1652515144	46646	40565	1397236905
keller6	1216523202	1423791481	62805	78042	1412480092
MANN_a27	1842521030	1751818471	116633	122709	356583240
MANN_a45	1416419765	1436860868	143450	244660	214877312
MANN_a81	1120079341	1130617479	1097	1082	88513354
MANN_a9	320505441	4521439	2061	1667	1216045
p_hat1000-1	195934761	25226263	105912	133502	16178765
p_hat1000-2	1419443679	1246580829	49553	37566	908264431
p_hat1000-3	1423870182	1103489502	23348	22326	872495041
p_hat1500-1	263689861	248019237	90212	95472	139333997
p_hat1500-2	1224605071	1108990759	72460	22400	955858496
p_hat1500-3	1217544104	1033953930	14804	14316	841237399
p_hat300-1	734645	131347	4193	4117	95753
p_hat300-2	1270967345	222733859	73029	73875	9781107
p_hat300-3	1400844719	1512408727	216429	214376	999105410
p_hat500-1	8507185	1258995	31095	31977	778165
p_hat500-2	1771576837	1200727352	118798	124476	968885358
p_hat500-3	1173860061	1116243883	85255	84451	954293027
p_hat700-1	39383251	4002661	60847	57775	3046223
p_hat700-2	1458963147	1124201031	63462	66353	846556017
p_hat700-3	1367119982	1038809377	43742	45037	929591245
san1000	1775991073	1786797847	28694	28295	1726549262
san200_0,7_1	3073877821	2995389744	81829	78415	1211889697
san200_0,7_2	3349553027	3237857891	47503	38365	2278672573
san200_0,9_1	1381626560	986615405	9793	8609	1262741727
san200_0,9_2	1393156189	1164293584	368894	359013	1305100820
san200_0,9_3	1355060699	1422714058	394340	374919	1230043269
san400_0,5_1	2689328254	2652935965	9827	9721	2149938440
san400_0,7_1	4799869803	4307336931	135343	138658	3188430388

Instance	nobound	basic	χ	χ + df	cp
brock200_1	792252455	876403943	491164	455516	258012511
san400_0,7_2	4814905618	4014250326	137632	137686	399780482
san400_0,7_3	2796941779	2509813178	152445	155739	2279564501
san400_0,9_1	1829460381	1280302308	113517	107745	1138385107
sanr200_0,7	1388680436	127231185	392881	406553	55196725
sanr200_0,9	1361280258	1250601843	371764	356436	1035046911
sanr400_0,5	831935923	55599271	238575	267951	35842055
sanr400_0,7	1819100184	1655058972	149595	152709	1449529234

Table 4: Number of branching steps generated by each algorithm for each instance. (Part 1 of 2)

Instance	df	dyn	mcq	mcr	mcs
brock200_1	45487955	457353	876403943	917233	810885
brock200_2	55593	7247	700185	7767	9475
brock200_3	696877	25375	8610083	32475	46007
brock200_4	2263155	94459	28636755	144115	106941
brock400_1	50871264	62013115	1570139627	64524246	1502283303
brock400_2	45171363	46140926	1488867444	68513460	1462157838
brock400_3	49531829	1286917370	1532761304	60811335	1509520441
brock400_4	50322425	45564505	1405415129	54905275	1402800673
brock800_1	54518463	58806772	1618560751	79209325	1640314702
brock800_2	49002281	59412938	1624630065	75197799	1599261656
brock800_3	49932338	62233087	1551138716	74193214	1558804796
brock800_4	51983014	50966767	1572500046	76024151	1560819523
c-fat200-1	39	437	811	377	377
c-fat200-2	85	487	1917	353	353
c-fat200-5	127	621	140789	285	285
c-fat500-1	205	1045	2151	973	973
c-fat500-10	321	1493	282162232	749	749
c-fat500-2	285	1093	8365	949	949
c-fat500-5	303	1245	571779	873	873
hamming10-2	1670366	2523	117563286	1025	1025
hamming10-4	65666598	21561034	1779980222	36526945	36480716
hamming6-2	23749	127	508981	65	65
hamming6-4	313	221	2091	165	159
hamming8-2	4571958	511	393187995	257	257
hamming8-4	4186763	41023	114637761	82985	53795
johnson16-2-4	6314291	1302089	36684707	646073	582951
johnson32-2-4	241752422	494069783	3016086146	483182549	350365121
johnson8-2-4	133	95	717	73	55
johnson8-4-4	23915	447	272001	289	345
keller4	1542373	17469	19055823	22311	41603
keller5	48418003	19746343	1571521075	23360773	21247054
keller6	53540849	8136128	1298588976	3081827	4071405
MANN_a27	141534916	76509	1732935128	76041	57228463
MANN_a45	103986423	168016	1435281442	199489	34339196
MANN_a81	77062739	18072	1079417006	20879	19734122
MANN_a9	590077	191	4521439	143	867
p_hat1000-1	1702829	340301	25226263	404185	362631
p_hat1000-2	26905625	12508436	1196720590	15765624	15887015
p_hat1000-3	25492655	11526969	1042770416	7767578	9636793
p_hat1500-1	15580561	2275349	247019237	2521053	2720917
p_hat1500-2	24579141	9742637	1044341143	6422538	6458550
p_hat1500-3	24327751	8084665	935694895	3208352	3018965
p_hat300-1	9463	4325	131347	4065	2793

Instance	df	dyn	mcq	mcr	mcs
p_hat300-2	5634539	15361	222733859	9979	86639
p_hat300-3	29784374	1251627	1166288803	3564775	19469638
p_hat500-1	89687	21687	1258995	21573	20341
p_hat500-2	28459444	388111	1141248373	856281	7270503
p_hat500-3	31466837	11502144	1083664046	17523386	16842610
p_hat700-1	226047	56073	4002661	68513	39325
p_hat700-2	25006718	2183587	1093780110	4843391	13873379
p_hat700-3	36952665	8813303	1012533803	8282230	7879094
san1000	48139216	251937	1704596479	454323	125941
san200_0,7_1	475486269	1681	2982284485	2641	3847
san200_0,7_2	360879277	3459	3203321315	3207	3209
san200_0,9_1	31138517	57917	991229807	335593	293955
san200_0,9_2	31339861	104463	1152027586	799783	1405665
san200_0,9_3	52015703	694687	1415136730	65169	2861673
san400_0,5_1	182535810	5845	2627347300	4013	1419
san400_0,7_1	601779262	85917	4251407753	212671	95731
san400_0,7_2	547888711	19579	4007433356	54853	1358825
san400_0,7_3	156782303	1111485	2443318218	830813	4924083
san400_0,9_1	237267123	752761	1275217628	227235	10169641
sanr200_0,7	9606435	210471	127231185	360205	338665
sanr200_0,9	34293738	12449743	1111389531	27375042	20157930
sanr400_0,5	4319331	500363	55599271	602511	500295
sanr400_0,7	57152713	52288774	1629072833	69227799	50360317

Table 5: Number of branching steps generated by each algorithm for each instance. (Part 2 of 2)

Instance	**nobound**	**basic**	χ	χ + **df**	**cp**
brock200_1	38400,00	22101,96	38400,08	38400,09	7016,14
brock200_2	362,10	14,49	544,24	489,45	9,69
brock200_3	21320,26	178,95	4653,89	4674,69	86,48
brock200_4	38400,00	663,04	9172,66	9504,07	374,27
brock400_1	38400,00	38400,00	38400,47	38400,25	38400,00
brock400_2	38400,00	38400,00	38400,09	38400,71	38400,00
brock400_3	38400,00	38400,00	38400,16	38400,07	38400,00
brock400_4	38400,00	38400,00	38400,38	38400,51	38400,00
brock800_1	38400,00	38400,00	38400,56	38400,10	38400,00
brock800_2	38400,00	38400,00	38400,75	38400,44	38400,00
brock800_3	38400,00	38400,00	38400,47	38400,94	38400,00
brock800_4	38400,00	38400,00	38400,51	38401,64	38400,00
c-fat200-1	4,62	0,08	3,14	3,11	0,10
c-fat200-2	5431,09	0,18	0,81	0,37	0,15
c-fat200-5	38400,00	9,64	23,86	17,56	38400,00
c-fat500-1	54,29	0,56	1,22	1,66	0,60
c-fat500-10	38400,00	38400,00	57,10	9,43	38400,00
c-fat500-2	38400,00	0,64	1,60	1,97	0,79
c-fat500-5	38400,00	32,93	11,05	2,34	1949,93
hamming10-2	38400,00	38400,00	3646,54	2561,53	38400,00
hamming10-4	38400,00	38400,00	38402,02	38402,52	38400,00
hamming6-2	38400,00	18,16	0,91	0,56	7,69
hamming6-4	0,06	0,03	1,42	1,12	0,03
hamming8-2	38400,00	38400,00	52,35	36,30	38400,00
hamming8-4	38400,00	2640,84	4036,40	5048,92	810,80

Instance	nobound	basic	χ	$\chi + \mathsf{df}$	cp
johnson16-2-4	1808,59	442,67	38400,01	38400,07	275,95
johnson32-2-4	38400,00	38400,00	38400,00	38401,18	38400,00
johnson8-2-4	0,02	0,01	0,21	0,27	0,01
johnson8-4-4	104,02	5,62	2,75	4,52	2,08
keller4	3555,98	330,91	3187,01	2988,21	144,65
keller5	38400,00	38400,00	38400,73	38401,48	38400,00
keller6	38400,00	38400,00	38400,40	38400,26	38400,00
MANN_a27	38400,00	38400,00	38400,27	38400,11	38400,00
MANN_a45	38400,00	38400,00	38400,24	38400,08	38400,00
MANN_a81	38400,00	38400,00	38400,14	38400,11	38400,00
MANN_a9	3953,11	72,72	19,81	17,75	20,15
p_hat1000-1	17965,73	555,75	38401,14	38400,37	345,07
p_hat1000-2	38400,00	38400,00	38400,81	38401,01	38400,00
p_hat1000-3	38400,00	38400,00	38403,18	38403,06	38400,00
p_hat1500-1	38400,00	5944,09	38400,93	38401,30	3382,21
p_hat1500-2	38400,00	38400,00	38405,42	38404,86	38400,00
p_hat1500-3	38400,00	38400,00	38405,08	38400,57	38400,00
p_hat300-1	28,87	2,69	239,68	191,77	1,92
p_hat300-2	38400,00	6554,11	9302,36	9180,40	311,92
p_hat300-3	38400,00	38400,00	38400,19	38400,21	38400,00
p_hat500-1	453,97	26,27	3714,26	3316,40	16,20
p_hat500-2	38400,00	38400,00	38400,07	38400,34	38400,00
p_hat500-3	38400,00	38400,00	38400,91	38400,55	38400,00
p_hat700-1	3432,44	97,95	13653,42	11739,93	65,96
p_hat700-2	38400,00	38400,00	38400,32	38400,73	38400,00
p_hat700-3	38400,00	38400,00	38400,66	38401,61	38400,00
san1000	38400,00	38400,00	38400,27	38400,12	38400,00
san200_0,7_1	38400,00	38400,00	6100,95	6071,77	38400,00
san200_0,7_2	38400,00	38400,00	3565,79	3017,89	38400,00
san200_0,9_1	38400,00	38400,00	1250,63	1237,34	38400,00
san200_0,9_2	38400,00	38400,00	38400,11	38400,28	38400,00
san200_0,9_3	38400,00	38400,00	38400,62	38400,26	38400,00
san400_0,5_1	38400,00	38400,00	2041,90	2069,92	38400,00
san400_0,7_1	38400,00	38400,00	38400,47	38400,04	38400,00
san400_0,7_2	38400,00	38400,00	38400,13	38400,21	38400,00
san400_0,7_3	38400,00	38400,00	38400,44	38400,19	38400,00
san400_0,9_1	38400,00	38400,00	38400,19	38400,30	38400,00
sanr200_0,7	38400,00	2995,25	28738,06	29998,11	38400,00
sanr200_0,9	38400,00	38400,00	38400,16	38400,12	38400,00
sanr400_0,5	38400,00	1109,32	38400,15	38400,36	746,72
sanr400_0,7	38400,00	38400,00	38400,15	38400,47	38400,00

Table 6: Running time, in seconds, of each algorithm over each instance. (Part 1 of 2)

Instance	**df**	**dyn**	**mcq**	**mcr**	**mcs**
brock200_1	38400,00	312,24	22520,15	492,99	610,93
brock200_2	32,38	3,20	14,73	5,26	6,43
brock200_3	415,40	12,53	189,24	17,04	25,75
brock200_4	1550,26	41,32	678,03	59,81	59,95
brock400_1	38400,00	38400,00	38400,00	38400,00	38400,00
brock400_2	38400,00	38400,00	38400,00	38400,00	38400,00
brock400_3	38400,00	38400,00	38400,00	38400,00	38400,00
brock400_4	38400,00	38400,00	38400,00	38400,00	38400,00
brock800_1	38400,00	38400,00	38400,00	38400,00	38400,00
brock800_2	38400,00	38400,00	38400,00	38400,00	38400,00
brock800_3	38400,00	38400,00	38400,00	38400,00	38400,00
brock800_4	38400,00	38400,00	38400,00	38400,00	38400,00
c-fat200-1	0,14	0,08	0,08	0,57	0,47
c-fat200-2	0,24	0,15	0,18	1,11	1,00
c-fat200-5	1,04	0,85	10,24	2,96	2,81
c-fat500-1	1,07	0,41	0,45	3,41	3,40
c-fat500-10	11,54	8,54	38400,00	37,86	37,95
c-fat500-2	1,29	0,54	0,58	6,41	6,41
c-fat500-5	3,49	2,03	44,75	16,83	16,80
hamming10-2	38400,24	527,18	38400,00	74,00	78,43
hamming10-4	38400,00	38400,00	38400,00	38400,00	38400,00
hamming6-2	37,12	0,07	18,57	0,07	0,83
hamming6-4	0,08	0,02	0,03	0,03	0,02
hamming8-2	38400,10	4,75	38400,00	1,29	1,30
hamming8-4	3584,62	29,56	2708,65	55,45	44,64
johnson16-2-4	981,24	118,53	442,22	49,11	58,29
johnson32-2-4	38400,00	38400,00	38400,00	38400,00	38400,00
johnson8-2-4	0,02	0,01	0,01	0,01	0,01
johnson8-4-4	12,21	0,17	5,60	0,20	0,18
keller4	614,28	7,30	342,04	9,42	17,59
keller5	38400,00	38400,00	38400,00	38400,00	38400,00
keller6	38400,00	38400,01	38400,00	38400,00	38400,01
MANN_a27	38400,00	2170,04	38400,00	1715,20	38400,01
MANN_a45	38400,00	38400,45	38400,00	38401,81	38400,00
MANN_a81	38400,00	38400,14	38400,00	38401,91	38400,01
MANN_a9	141,08	0,05	73,97	0,08	0,35
p_hat1000-1	1255,99	154,95	592,01	313,79	404,89
p_hat1000-2	38400,00	38400,00	38400,00	38400,00	38400,00
p_hat1000-3	38400,00	38400,00	38400,00	38400,01	38400,02
p_hat1500-1	13254,73	1060,08	6299,25	1650,19	2450,28
p_hat1500-2	38400,01	38400,01	38400,00	38400,00	38400,09
p_hat1500-3	38400,00	38400,02	38400,00	38400,02	38400,00
p_hat300-1	6,13	0,81	2,70	5,65	5,80
p_hat300-2	7295,30	12,91	6851,90	18,16	115,22
p_hat300-3	38400,00	2126,53	38400,00	4560,07	38400,00
p_hat500-1	57,51	5,92	27,87	30,41	32,39
p_hat500-2	38400,00	616,11	38400,00	1171,40	14987,52
p_hat500-3	38400,00	38400,00	38400,00	38400,00	38400,01
p_hat700-1	213,46	31,87	101,34	87,85	89,93
p_hat700-2	38400,00	5868,05	38400,00	10536,45	38400,00
p_hat700-3	38400,00	38400,01	38400,00	38400,01	38400,00
san1000	38400,00	358,26	38400,00	3642,93	612,66
san200_0,7_1	38400,00	2,46	38400,00	6,63	7,13
san200_0,7_2	38400,00	2,78	38400,00	4,11	6,09
san200_0,9_1	38400,01	129,38	38400,00	431,70	913,97
san200_0,9_2	38400,02	301,93	38400,00	948,02	2502,45
san200_0,9_3	38400,00	1712,41	38400,00	96,48	8561,04

Instance	df	dyn	mcq	mcr	mcs
san400_0,5_1	38400,00	6,58	38400,00	24,98	19,26
san400_0,7_1	38400,00	163,05	38400,00	571,20	274,31
san400_0,7_2	38400,00	63,86	38400,00	139,88	1755,94
san400_0,7_3	38400,00	889,02	38400,00	1067,32	4930,04
san400_0,9_1	38400,00	10398,81	38400,00	2728,54	38400,01
sanr200_0,7	6743,79	113,64	2996,13	148,32	194,69
sanr200_0,9	38400,00	24012,09	38400,00	38400,00	38400,00
sanr400_0,5	2398,07	184,34	1140,04	209,96	250,37
sanr400_0,7	38400,01	38400,00	38400,00	38400,00	38400,00

Table 7: Running time, in seconds, of each algorithm over each instance. (Part 2 of 2)

Instance	nobound	basic	χ	χ+df
brock200_1	4,85E-05	2,52E-05	7,82E-02	8,43E-02
brock200_2	2,88E-05	2,07E-05	5,41E-02	4,93E-02
brock200_3	5,83E-05	2,08E-05	6,25E-02	6,26E-02
brock200_4	5,87E-05	2,32E-05	6,94E-02	7,20E-02
brock400_1	2,80E-05	2,43E-05	2,68E-01	2,98E-01
brock400_2	2,79E-05	2,52E-05	2,75E-01	2,75E-01
brock400_3	2,47E-05	2,49E-05	2,88E-01	2,75E-01
brock400_4	2,81E-05	2,58E-05	2,89E-01	2,80E-01
brock800_1	2,67E-05	2,26E-05	3,96E-01	3,71E-01
brock800_2	2,73E-05	2,33E-05	4,68E-01	3,96E-01
brock800_3	2,56E-05	2,37E-05	4,21E-01	3,54E-01
brock800_4	2,57E-05	2,37E-05	4,47E-01	3,34E-01
c-fat200-1	2,84E-05	1,03E-04	5,33E-02	7,99E-02
c-fat200-2	2,82E-05	9,26E-05	1,65E-02	7,33E-02
c-fat200-5	2,82E-05	6,84E-05	9,98E-02	1,38E-01
c-fat500-1	4,82E-05	2,58E-04	4,19E-02	3,33E-01
c-fat500-10	1,08E-04	1,34E-04	2,26E-01	1,89E+00
c-fat500-2	4,92E-05	7,60E-05	3,02E-02	3,94E-01
c-fat500-5	5,31E-05	5,76E-05	8,57E-02	4,68E-01
hamming10-2	2,04E-04	3,25E-04	3,56E+00	4,72E+00
hamming10-4	2,06E-05	2,08E-05	1,01E+00	9,67E-01
hamming6-2	2,52E-05	3,57E-05	1,40E-02	1,69E-02
hamming6-4	1,60E-05	1,66E-05	7,11E-03	6,27E-03
hamming8-2	5,99E-05	9,65E-05	2,04E-01	2,81E-01
hamming8-4	3,01E-05	2,30E-05	1,18E-01	1,76E-01
johnson16-2-4	1,96E-05	1,21E-05	2,02E-02	1,98E-02
johnson32-2-4	1,12E-05	1,23E-05	5,88E-02	4,94E-02
johnson8-2-4	1,19E-05	1,27E-05	2,55E-03	3,39E-03
johnson8-4-4	1,35E-05	2,07E-05	1,95E-02	3,45E-02
keller4	2,79E-05	1,74E-05	4,95E-02	4,97E-02
keller5	2,67E-05	2,32E-05	8,23E-01	9,47E-01
keller6	3,16E-05	2,70E-05	6,11E-01	4,92E-01
MANN_a27	2,08E-05	2,19E-05	3,29E-01	3,13E-01
MANN_a45	2,71E-05	2,67E-05	2,68E-01	1,57E-01
MANN_a81	3,43E-05	3,40E-05	6,65E+01	7,98E+01
MANN_a9	1,23E-05	1,61E-05	9,61E-03	1,06E-02
p_hat1000-1	9,17E-05	2,20E-05	3,63E-01	2,88E-01
p_hat1000-2	2,71E-05	3,08E-05	7,75E-01	1,02E+00
p_hat1000-3	2,70E-05	3,48E-05	1,64E+00	1,72E+00
p_hat1500-1	1,46E-04	2,40E-05	4,26E-01	4,02E-01
p_hat1500-2	3,14E-05	3,46E-05	5,30E-01	1,71E+00
p_hat1500-3	3,15E-05	3,71E-05	2,59E+00	2,68E+00

Instance	**nobound**	**basic**	χ	χ**+df**
p_hat300-1	3,93E-05	2,05E-05	5,72E-02	4,66E-02
p_hat300-2	3,02E-05	2,94E-05	1,27E-01	1,24E-01
p_hat300-3	2,74E-05	2,54E-05	1,77E-01	1,79E-01
p_hat500-1	5,34E-05	2,09E-05	1,19E-01	1,04E-01
p_hat500-2	2,17E-05	3,20E-05	3,23E-01	3,08E-01
p_hat500-3	3,27E-05	3,44E-05	4,50E-01	4,55E-01
p_hat700-1	8,72E-05	2,45E-05	2,24E-01	2,03E-01
p_hat700-2	2,63E-05	3,42E-05	6,05E-01	5,79E-01
p_hat700-3	2,81E-05	3,70E-05	8,78E-01	8,53E-01
san1000	2,16E-05	2,15E-05	1,34E+00	1,36E+00
san200_0,7_1	1,25E-05	1,28E-05	7,46E-02	7,74E-02
san200_0,7_2	1,15E-05	1,19E-05	7,51E-02	7,87E-02
san200_0,9_1	2,78E-05	3,89E-05	1,28E-01	1,44E-01
san200_0,9_2	2,76E-05	3,30E-05	1,04E-01	1,07E-01
san200_0,9_3	2,83E-05	2,70E-05	9,74E-02	1,02E-01
san400_0,5_1	1,43E-05	1,45E-05	2,08E-01	2,13E-01
san400_0,7_1	8,00E-06	8,92E-06	2,84E-01	2,77E-01
san400_0,7_2	7,98E-06	9,57E-06	2,79E-01	2,79E-01
san400_0,7_3	1,37E-05	1,53E-05	2,52E-01	2,47E-01
san400_0,9_1	2,10E-05	3,00E-05	3,38E-01	3,56E-01
sanr200_0,7	2,77E-05	2,35E-05	7,31E-02	7,38E-02
sanr200_0,9	2,82E-05	3,07E-05	1,03E-01	1,08E-01
sanr400_0,5	4,62E-05	2,00E-05	1,61E-01	1,43E-01
sanr400_0,7	2,11E-05	2,32E-05	2,57E-01	2,51E-01

Table 8: Time, in seconds, that each algorithm spent on each branching step. (Part 1 of 2)

Instance	cp	df	dyn	mcq	mcr	mcs
brock200_1	2,72E-05	8,44E-04	6,83E-04	2,57E-05	5,37E-04	7,53E-04
brock200_2	2,13E-05	5,82E-04	4,41E-04	2,10E-05	6,77E-04	6,79E-04
brock200_3	2,41E-05	5,96E-04	4,94E-04	2,20E-05	5,25E-04	5,60E-04
brock200_4	2,37E-05	6,85E-04	4,37E-04	2,37E-05	4,15E-04	5,61E-04
brock400_1	2,72E-05	7,55E-04	6,19E-04	2,45E-05	5,95E-04	2,56E-05
brock400_2	2,74E-05	8,50E-04	8,32E-04	2,58E-05	5,60E-04	2,63E-05
brock400_3	2,86E-05	7,75E-04	2,98E-05	2,51E-05	6,31E-04	2,54E-05
brock400_4	2,73E-05	7,63E-04	8,43E-04	2,73E-05	6,99E-04	2,74E-05
brock800_1	2,47E-05	7,04E-04	6,53E-04	2,37E-05	4,85E-04	2,34E-05
brock800_2	2,42E-05	7,84E-04	6,46E-04	2,36E-05	5,11E-04	2,40E-05
brock800_3	2,50E-05	7,69E-04	6,17E-04	2,48E-05	5,18E-04	2,46E-05
brock800_4	2,58E-05	7,39E-04	7,53E-04	2,44E-05	5,05E-04	2,46E-05
c-fat200-1	1,88E-04	3,70E-03	1,74E-04	1,04E-04	1,50E-03	1,25E-03
c-fat200-2	6,32E-05	2,82E-03	3,07E-04	9,44E-05	3,15E-03	2,84E-03
c-fat200-5	5,96E-05	8,16E-03	1,37E-03	7,28E-05	1,04E-02	9,87E-03
c-fat500-1	5,04E-04	5,22E-03	3,89E-04	2,08E-04	3,50E-03	3,49E-03
c-fat500-10	1,11E-04	3,59E-02	5,72E-03	1,36E-04	5,06E-02	5,07E-02
c-fat500-2	1,87E-04	4,53E-03	4,90E-04	6,94E-05	6,75E-03	6,75E-03
c-fat500-5	6,53E-05	1,15E-02	1,63E-03	7,83E-05	1,93E-02	1,92E-02
hamming10-2	3,45E-04	2,30E-02	2,09E-01	3,27E-04	7,22E-02	7,65E-02
hamming10-4	2,30E-05	5,85E-04	1,78E-03	2,16E-05	1,05E-03	1,05E-03
hamming6-2	3,77E-05	1,56E-03	5,33E-04	3,65E-05	1,14E-03	1,27E-02
hamming6-4	1,77E-05	2,66E-04	9,60E-05	1,63E-05	1,77E-04	1,50E-04
hamming8-2	9,93E-05	8,40E-03	9,29E-03	9,77E-05	5,00E-03	5,07E-03
hamming8-4	2,39E-05	8,56E-04	7,20E-04	2,36E-05	6,68E-04	8,30E-04

Instance	cp	df	dyn	mcq	mcr	mcs
johnson16-2-4	1,19E-05	1,55E-04	9,10E-05	1,21E-05	7,60E-05	1,00E-04
johnson32-2-4	1,25E-05	1,59E-04	7,77E-05	1,27E-05	7,95E-05	1,10E-04
johnson8-2-4	1,75E-05	1,53E-04	6,00E-05	1,26E-05	7,84E-05	1,08E-04
johnson8-4-4	2,38E-05	5,11E-04	3,90E-04	2,06E-05	6,83E-04	5,26E-04
keller4	1,70E-05	3,98E-04	4,18E-04	1,79E-05	4,22E-04	4,23E-04
keller5	2,75E-05	7,93E-04	1,94E-03	2,44E-05	1,64E-03	1,81E-03
keller6	2,72E-05	7,17E-04	4,72E-03	2,96E-05	1,25E-02	9,43E-03
MANN_a27	1,08E-04	2,71E-04	2,84E-02	2,22E-05	2,26E-02	6,71E-04
MANN_a45	1,79E-04	3,69E-04	2,29E-01	2,68E-05	1,93E-01	1,12E-03
MANN_a81	4,34E-04	4,98E-04	2,12E+00	3,56E-05	1,84E+00	1,95E-03
MANN_a9	1,66E-05	2,39E-04	2,86E-04	1,64E-05	5,30E-04	4,08E-04
p_hat1000-1	2,13E-05	7,38E-04	4,55E-04	2,35E-05	7,76E-04	1,12E-03
p_hat1000-2	4,23E-05	1,43E-03	3,07E-03	3,21E-05	2,44E-03	2,42E-03
p_hat1000-3	4,40E-05	1,51E-03	3,33E-03	3,68E-05	4,94E-03	3,98E-03
p_hat1500-1	2,43E-05	8,51E-04	4,66E-04	2,55E-05	6,55E-04	9,01E-04
p_hat1500-2	4,02E-05	1,56E-03	3,94E-03	3,68E-05	5,98E-03	5,95E-03
p_hat1500-3	4,56E-05	1,58E-03	4,75E-03	4,10E-05	1,20E-02	1,27E-02
p_hat300-1	2,00E-05	6,47E-04	1,87E-04	2,06E-05	1,39E-03	2,08E-03
p_hat300-2	3,19E-05	1,29E-03	8,40E-04	3,08E-05	1,82E-03	1,33E-03
p_hat300-3	3,84E-05	1,29E-03	1,70E-03	3,29E-05	1,28E-03	1,97E-03
p_hat500-1	2,08E-05	6,41E-04	2,73E-04	2,21E-05	1,41E-03	1,59E-03
p_hat500-2	3,96E-05	1,35E-03	1,59E-03	3,36E-05	1,37E-03	2,06E-03
p_hat500-3	4,02E-05	1,22E-03	3,34E-03	3,54E-05	2,19E-03	2,28E-03
p_hat700-1	2,17E-05	9,44E-04	5,68E-04	2,53E-05	1,28E-03	2,29E-03
p_hat700-2	4,54E-05	1,54E-03	2,69E-03	3,51E-05	2,18E-03	2,77E-03
p_hat700-3	4,13E-05	1,04E-03	4,36E-03	3,79E-05	4,64E-03	4,87E-03
san1000	2,22E-05	7,98E-04	1,42E-03	2,25E-05	8,02E-03	4,86E-03
san200_0,7_1	3,17E-05	8,08E-05	1,47E-03	1,29E-05	2,51E-03	1,85E-03
san200_0,7_2	1,69E-05	1,06E-04	8,03E-04	1,20E-05	1,28E-03	1,90E-03
san200_0,9_1	3,04E-05	1,23E-03	2,23E-03	3,87E-05	1,29E-03	3,11E-03
san200_0,9_2	2,94E-05	1,23E-03	2,89E-03	3,33E-05	1,19E-03	1,78E-03
san200_0,9_3	3,12E-05	7,38E-04	2,47E-03	2,71E-05	1,48E-03	2,99E-03
san400_0,5_1	1,79E-05	2,10E-04	1,13E-03	1,46E-05	6,22E-03	1,36E-02
san400_0,7_1	1,20E-05	6,38E-05	1,90E-03	9,03E-06	2,69E-03	2,87E-03
san400_0,7_2	9,61E-05	7,01E-05	3,26E-03	9,58E-06	2,55E-03	1,29E-03
san400_0,7_3	1,68E-05	2,45E-04	8,00E-04	1,57E-05	1,28E-03	1,00E-03
san400_0,9_1	3,37E-05	1,62E-04	1,38E-02	3,01E-05	1,20E-02	3,78E-03
sanr200_0,7	6,96E-04	7,02E-04	5,40E-04	2,35E-05	4,12E-04	5,75E-04
sanr200_0,9	3,71E-05	1,12E-03	1,93E-03	3,46E-05	1,40E-03	1,90E-03
sanr400_0,5	2,08E-05	5,55E-04	3,68E-04	2,05E-05	3,48E-04	5,00E-04
sanr400_0,7	2,65E-05	6,72E-04	7,34E-04	2,36E-05	5,55E-04	7,63E-04

Table 9: Time, in seconds, that each algorithm spent on each branching step. (Part 2 of 2)

References

[BC96] Mark Brockington and Joseph C Culberson, *Camouflaging independent sets in quasi-random graphs*, Cliques, coloring, and satisfiability: Second DIMACS implementation challenge **26** (1996), 75–88.

[BP90] P. Berman and A. Pelc, *Distributed probabilistic fault diagnosis for multiprocessor systems*, 20th Int. Symp. on Fault-Tolerant Computing Systems (FTCS-20) (1990), 340–346.

[CP90] Randy Carraghan and Panos M. Pardalos, *An exact algorithm for the maximum clique problem*, Operations Research Letters **9** (1990), no. 6, 375–382.

[CS90] K. Corrádi and S. Szabó, *A combinatorial approach for Keller's conjecture*, Periodica Mathematica Hungarica **21** (1990), no. 2, 95–100.

[CZ12] Renato Carmo and Alexandre Züge, *Branch and bound algorithms for the maximum clique problem under a unified framework*, Journal of the Brazilian Computer Society **18** (2012), no. 2, 137–151.

[dA15] Cleverson Sebastião dos Anjos, *Análise experimental de algoritmos*, Master's thesis, Federal University of Paraná, 2015.

[Fah02] Torsten Fahle, *Simple and fast: Improving a branch-and-bound algorithm for maximum clique*, Algorithms ESA 2002 **2461** (2002), 485–498.

[JAMS91] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon, *Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning*, Operations Research **39** (1991), no. 3, 378–406.

[JT96] David J. Johnson and Michael A. Trick (eds.), *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge, workshop, october 11-13, 1993*, American Mathematical Society, Boston, MA, USA, 1996.

[KJ07] Janez Konc and Dušanka Janezic, *An improved branch and bound algorithm for the maximum clique problem*, MATCH Commun. Math. Comput. Chem. **58(3)** (2007), 569–590.

[LS92] Jeffrey C. Lagarias and Peter W. Shor, *Keller's cube-tiling conjecture is false in high dimensions*, BAMS: Bulletin of the American Mathematical Society **27** (1992), no. 2, 279–283.

[McG12] Catherine C. McGeoch, *A guide to experimental algorithmics*, 1st ed., Cambridge University Press, New York, NY, USA, 2012.

[S+15] W. A. Stein et al., *Sage Mathematics Software (Version 6.5)*, The Sage Development Team, 2015, `http://www.sagemath.org`.

[San92] Laura A. Sanchis, *Generating test cases for the vertex cover problem*, Tech. Report, DIMACS, March 1992.

[TK07] Etsuji Tomita and Toshikatsu Kameda, *An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments*, Journal of Global Optimization **37** (2007), no. 1, 95–111.

[TS03] Etsuji Tomita and Tomokazu Seki, *An efficient branch-and-bound algorithm for finding a maximum clique*, Discrete Mathematics and Theoretical Computer Science, Proceedings of the 4th international conference on Discrete mathematics and theoretical computer science (2003), 278–289.

[TSH+10] Etsuji Tomita, Yoichi Sutani, Takanori Higashi, Shinya Takahashi, and Mitsuo Wakatsuki, *A simple and faster branch-and-bound algorithm for finding a maximum clique*, WALCOM: Algorithms and Computation **5942** (2010), 191–203.

[Zï1] Alexandre Züge, *Solução exata do problema da clique máxima*, Master's thesis, Federal University of Paraná, 2011.

Cleverson Sebastião dos Anjos
Depto. de Informática, UFPR
Brazil
cleverson.dosanjos@uol.com.br

Alexandre Prusch Züge
Depto. de Informática, UFPR
Brazil
alexandrezuge@gmail.com

Renato Carmo
Depto. de Informática, UFPR
Brazil
renato.carmo.rc@gmail.com