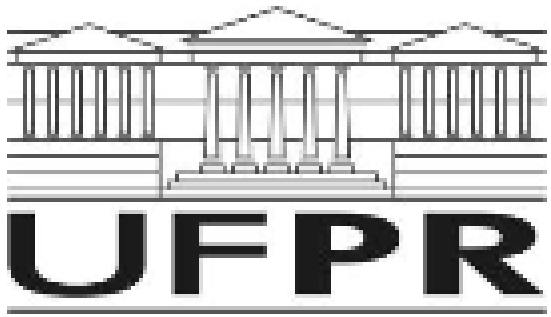
Worst Case Instances for Maximum Clique Algorithms and How to Avoid Them

Alexandre Prusch Züge Renato Carmo

Universidade Federal do Paraná
Programa de Pós-Graduação em Informática
Grupo de Pesquisa em Algoritmos

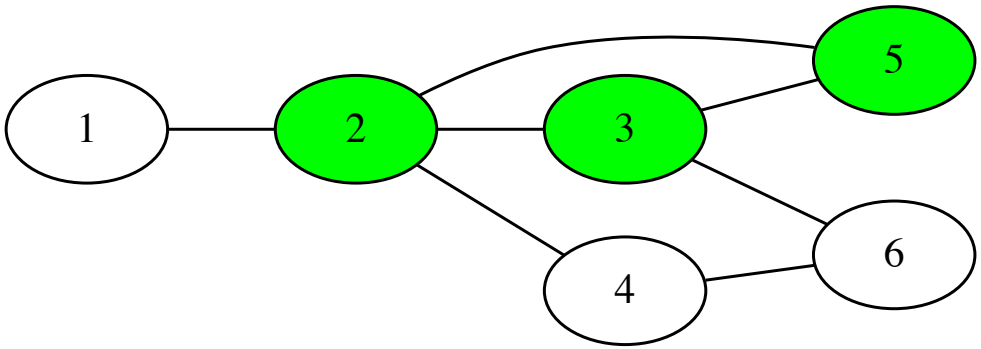
November 10, 2016



Introduction

Maximum Clique Problem (MC)
Input : Simple graph G.
Output : Maximum clique in G.

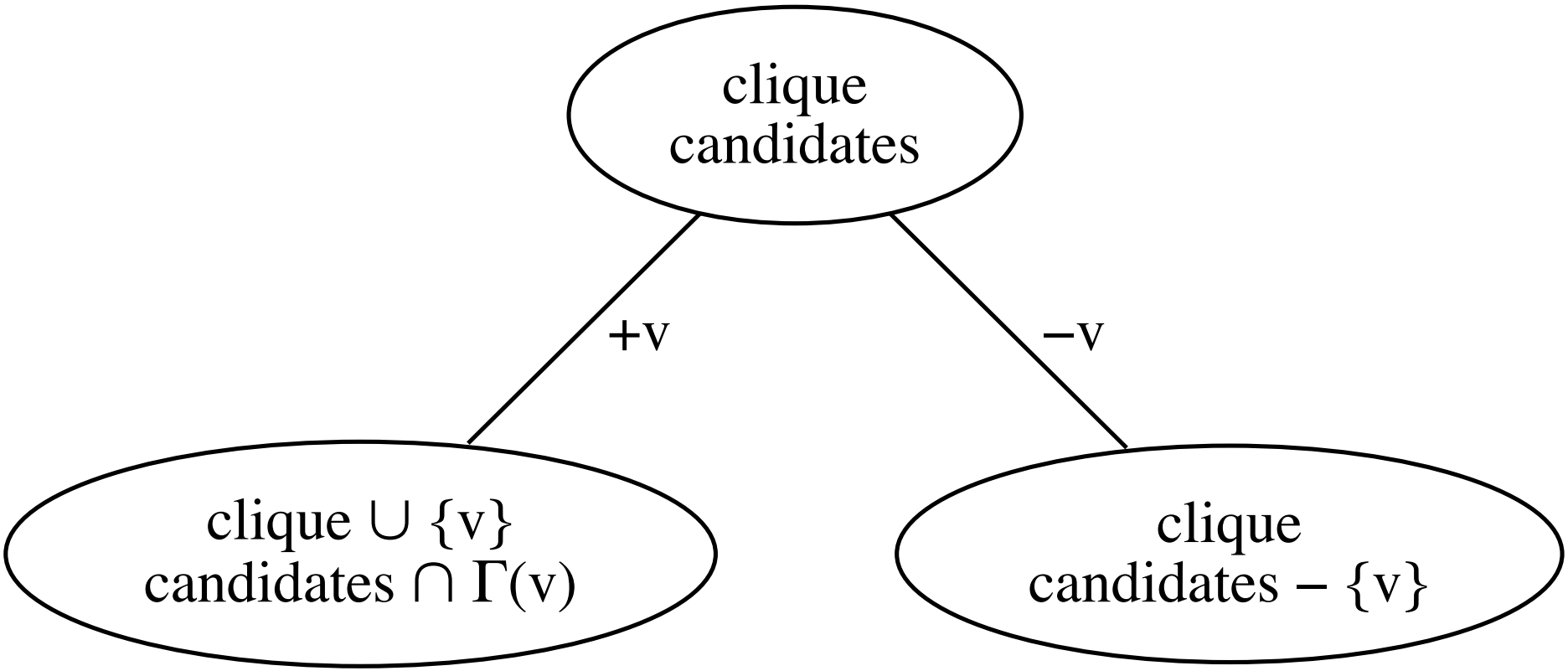
Maximum clique: $\{2, 3, 5\}$.



Clique: set of vertices inducing a complete subgraph.

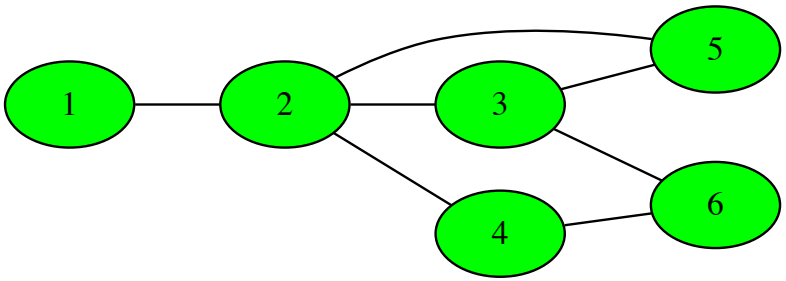
Branch and Bound Scheme for MC

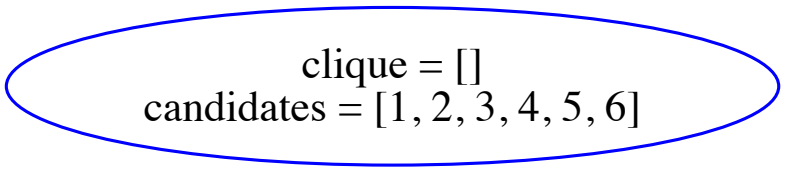
Generating cliques via backtracking.



Example

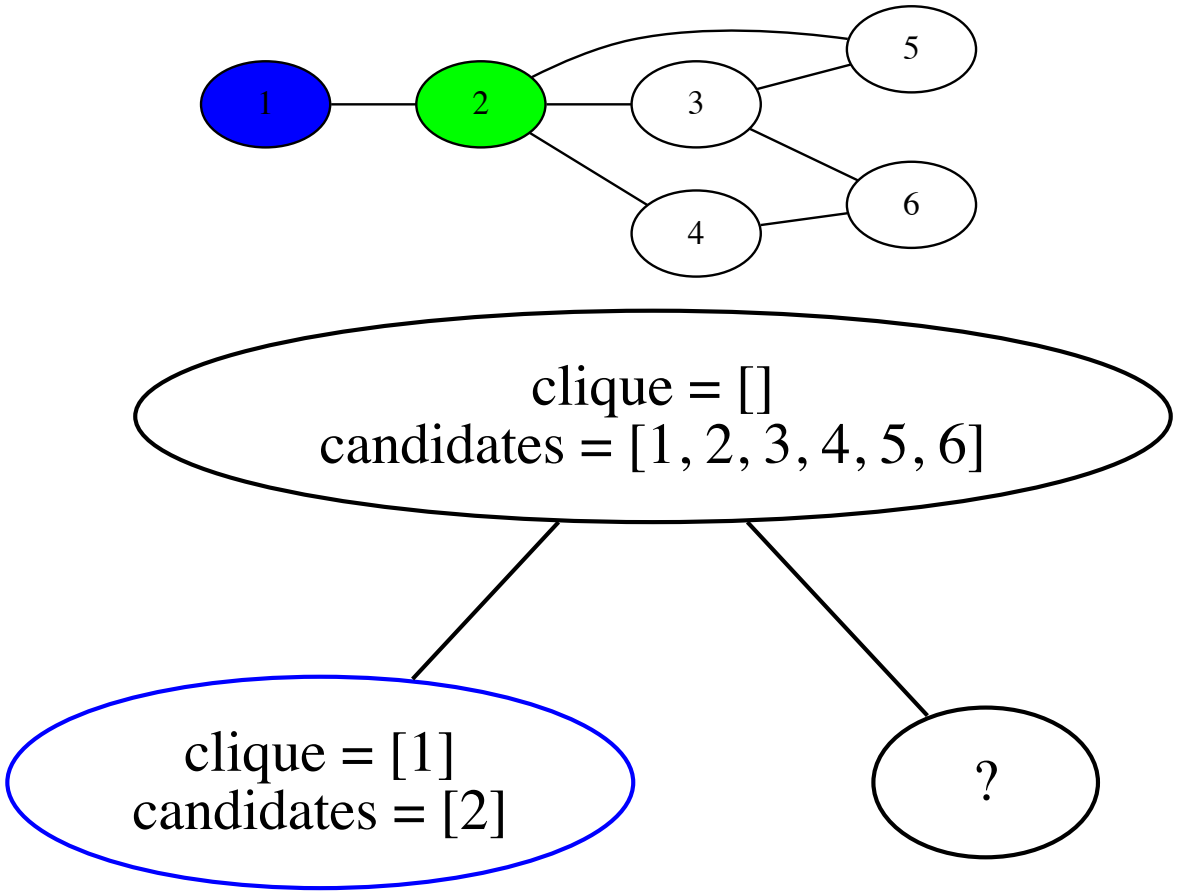
A graph and the search space.





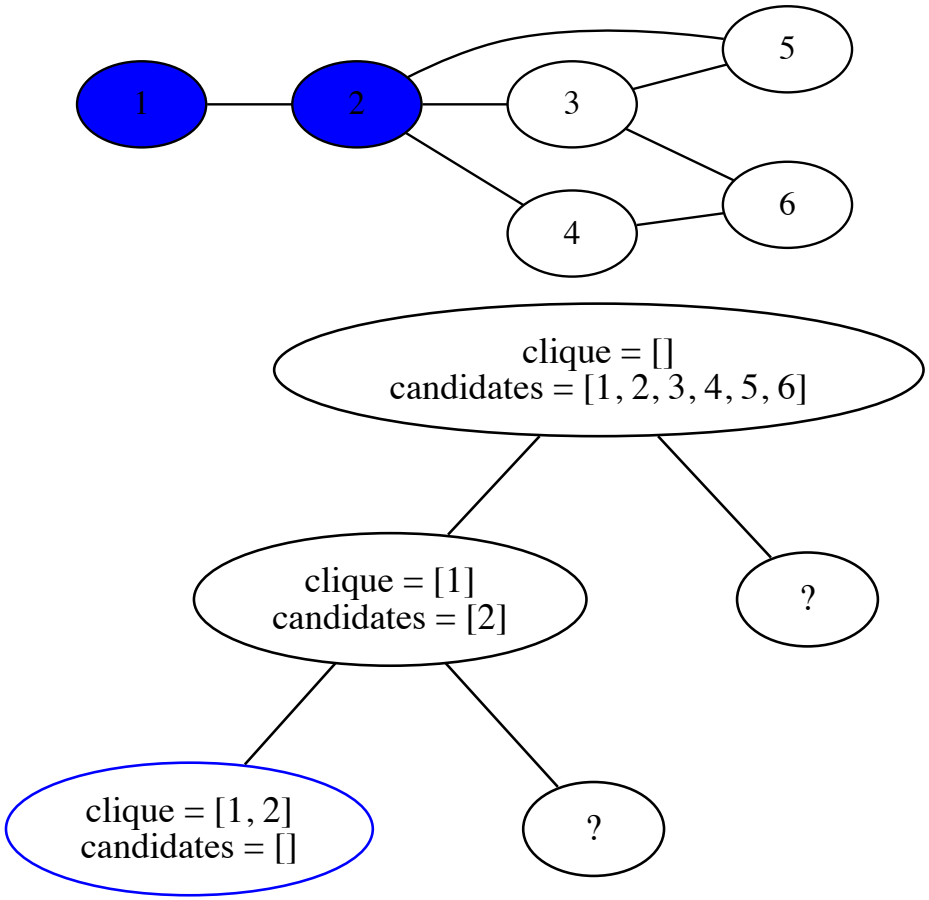
Example

A graph and the search space.



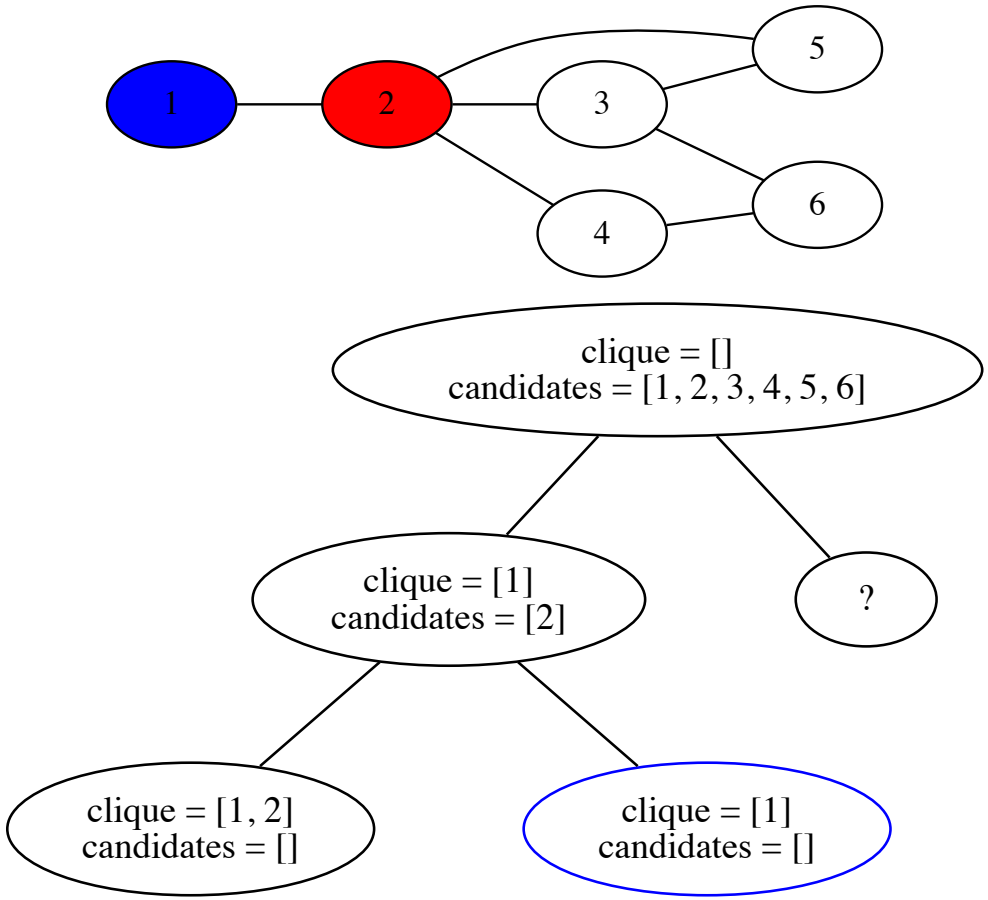
Example

A graph and the search space.



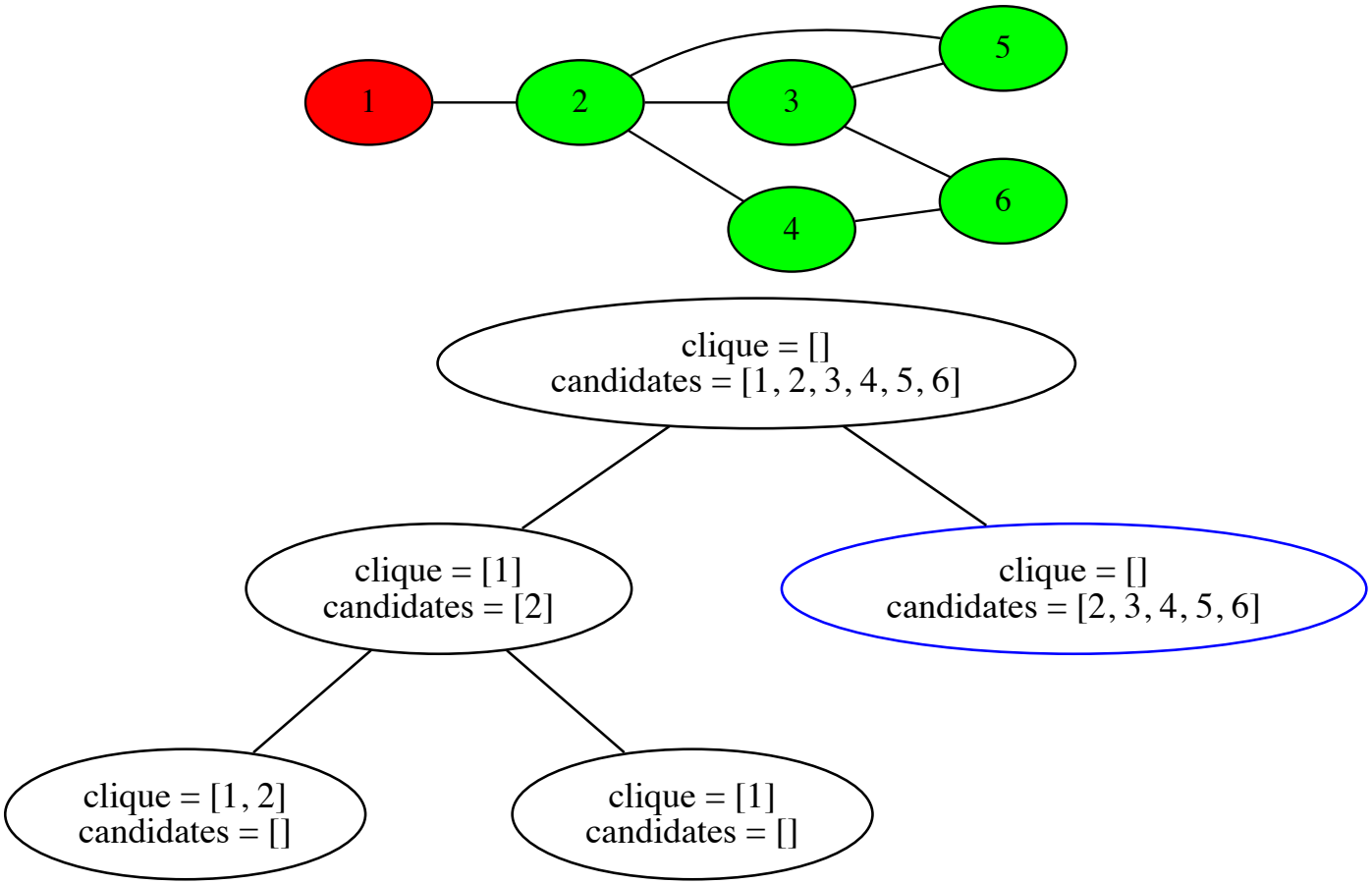
Example

A graph and the search space.



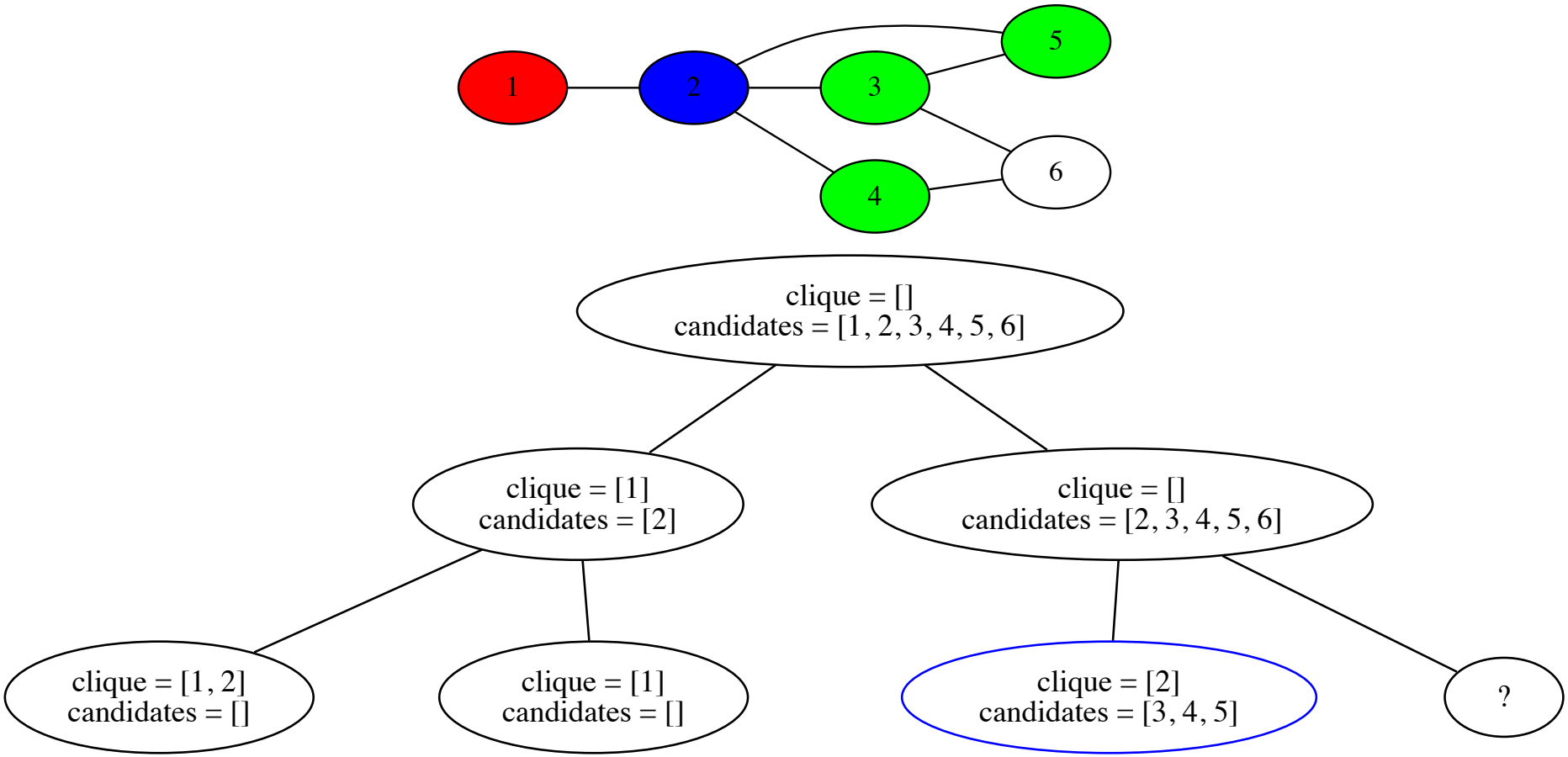
Example

A graph and the search space.



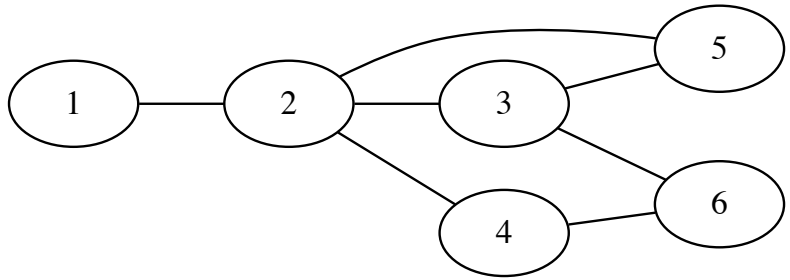
Example

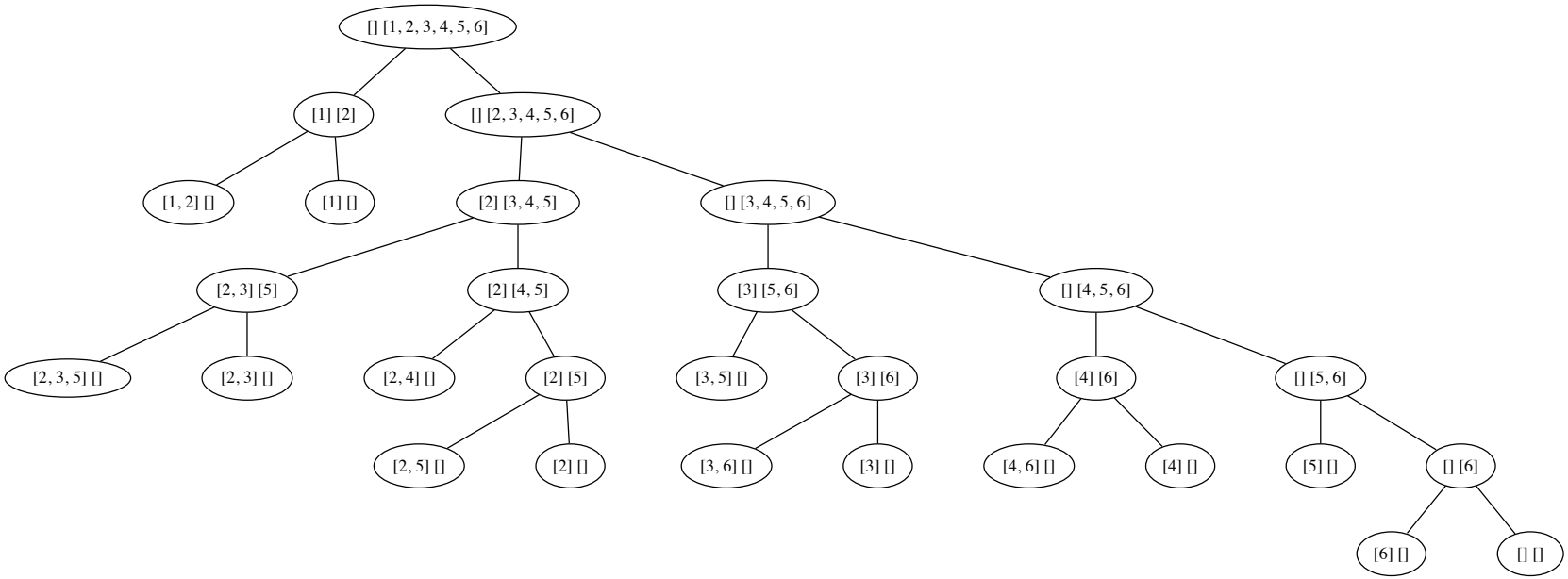
A graph and the search space.



Example

A graph and the search space (29 states in total).





Branch and Bound Scheme – Algorithm

MaxClique(G, Q, K, C)

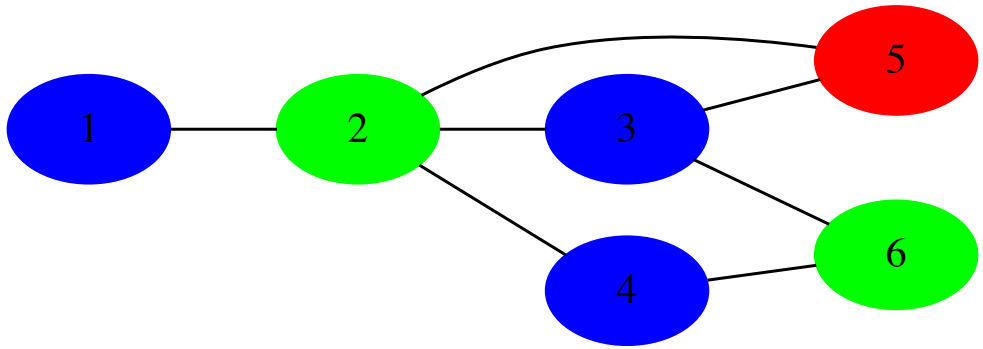
Input : a graph G, a clique Q in G, a set K of vertices in G and a clique C in G.

Output : a maximum clique in $G[Q \cup K]$ containing Q, or C, whichever is larger.

```
1  pre–process the graph G and the sets Q and K
2  if K = ∅ then
3  |   if |Q| > |C| then
4  |   |   C ← Q
5  else
6  |   if |Q| + upper-bound(G, K) > |C| then
7  |   |   remove a vertex v from K
8  |   |   C ← MaxClique(G, Q ∪ {v}, K ∩ Γ_G(v), C)
9  |   |   C ← MaxClique(G, Q, K − {v}, C)
10 return C
```

Coloring of G

Coloring $\gamma(G)$: partition of $V(G)$ into *colors* (independent sets).

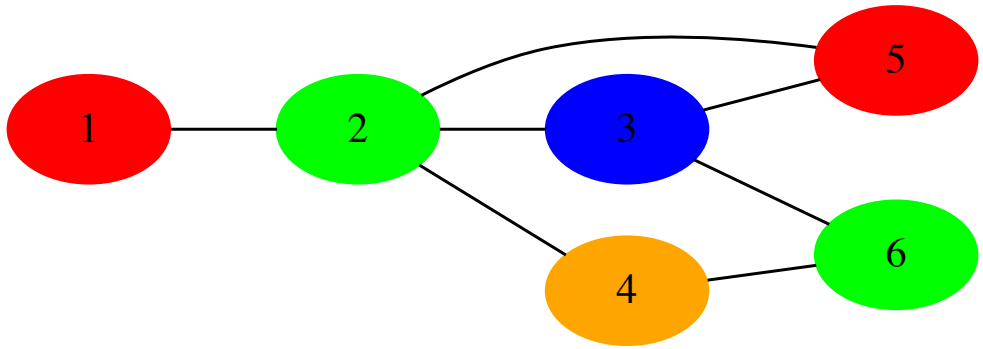


Number of colors is an upper bound for maximum clique:

$$\omega(G) \leq \chi(G)$$

Coloring of G

Coloring $\gamma(G)$: partition of $V(G)$ into *colors* (independent sets).



Number of colors is an upper bound for maximum clique:

$$\omega(G) \leq \chi(G) \leq |\gamma(G)|$$

Some Background

- MC is $\mathcal{NP}$–hard
- approximation to within a factor of $O(n^{1-\varepsilon})$ (n: number of vertices) remains $\mathcal{NP}$–hard for all $\varepsilon > 0$

Some Background

- MC is $\mathcal{NP}$–hard
- approximation to within a factor of $O(n^{1-\varepsilon})$ (n: number of vertices) remains $\mathcal{NP}$–hard for all $\varepsilon > 0$
- However...
  - Several algorithms for MC are reported to solve instances of practical interest and considerable size.
  - The best algorithms are based on the branch and bound technique and apply coloring for bounding.
  - From an experimental point of view, these algorithms display a behavior that seems sub-exponential.

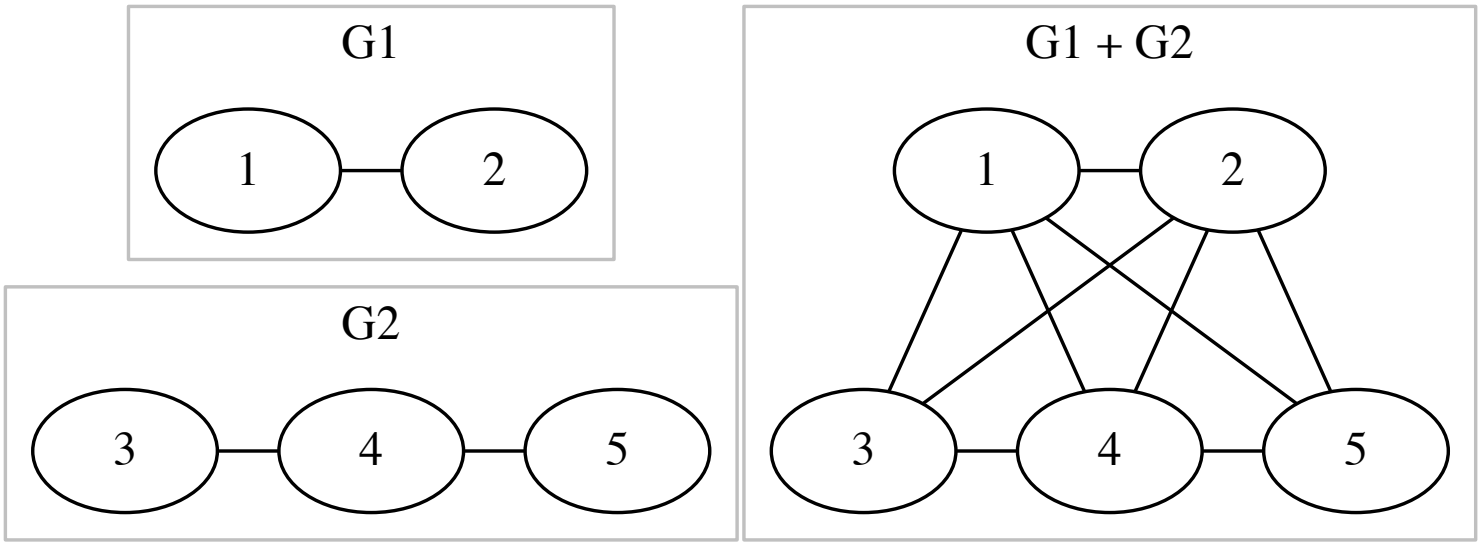
Some Background

- Very little is known about worst case instances for these algorithms.
- qC_5: infinite class of graphs for which these algorithms must take an exponential ($\Omega(2^{n/5})$) number of steps (Lavnikevich, 2013).
  - Based on *join graphs*.

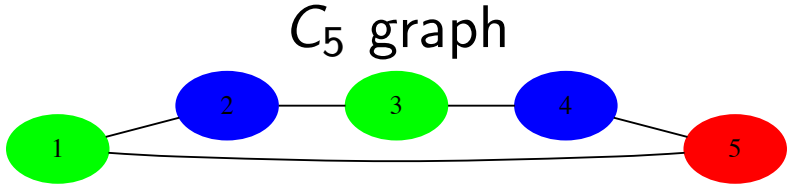
Graph Join

Given two disjoint graphs G_1 and G_2, the *join* of G_1 and G_2 is the graph $G_1 + G_2$ such that

$$V(G_1 + G_2) = V(G_1) \cup V(G_2)$$
$$E(G_1 + G_2) = E(G_1) \cup E(G_2) \cup \{\{v_1, v_2\} \mid v_1 \in V(G_1), v_2 \in V(G_2)\}$$



A Lower Bound for the Worst Case



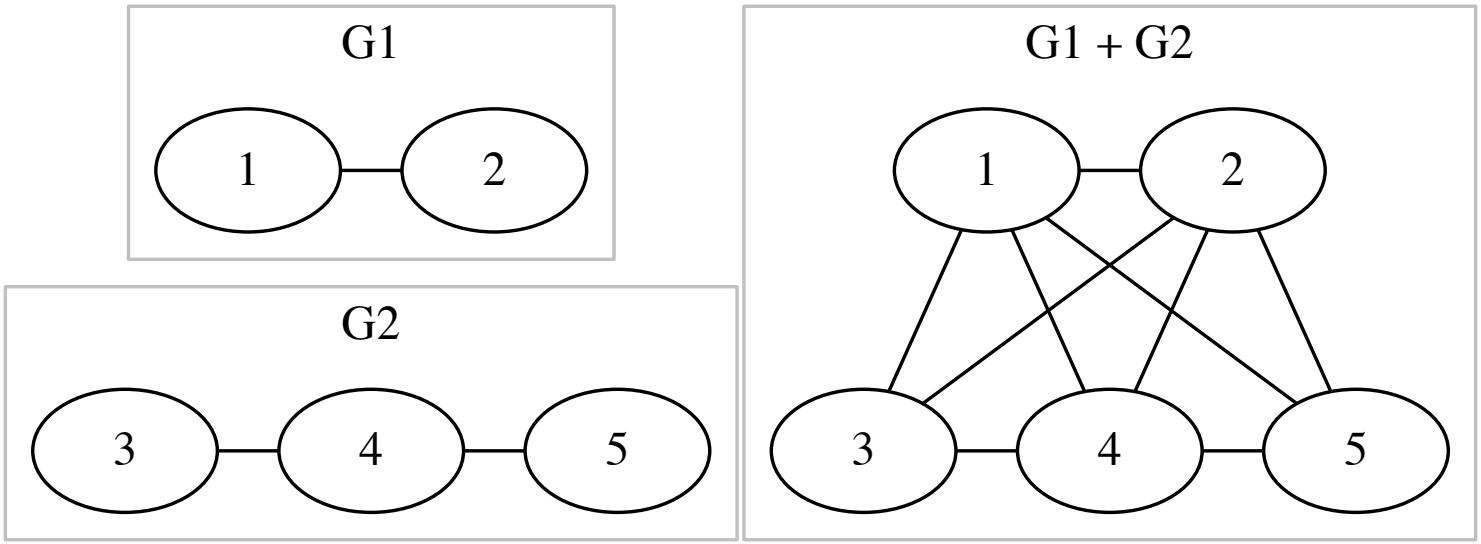
qC_5: graph join of q copies of C_5

$$|V(qC_5)| = 5q$$
$$\omega(qC_5) = 2q$$
$$\chi(qC_5) = 3q$$

More on Join Graphs

A *join graph* is a graph obtained by joining two graphs. For a graph $G = G_1 + G_2$, we shall call G_1 and G_2 *parts* of G.
Let $\mathcal{C}(G)$ be a maximum clique in G, then

$$V(G_1 + G_2) = V(G_1) \cup V(G_2)$$
$$\mathcal{C}(G_1 + G_2) = \mathcal{C}(G_1) \cup \mathcal{C}(G_2)$$



First Step – solving MC on a join graph

Let $G = G_1 + G_2$. Instead of finding a maximum clique in G, it suffices to find maximum cliques in G_1 and G_2 and make the union at the end.

First Step – solving MC on a join graph

Let $G = G_1 + G_2$. Instead of finding a maximum clique in G, it suffices to find maximum cliques in G_1 and G_2 and make the union at the end.
More generally, if a graph is the join of several graphs, we can find the maximum clique by finding the maximum cliques for each part.
$\mathcal{C}(G_1 + G_2 + \ldots + G_k) = \mathcal{C}(G_1) \cup \mathcal{C}(G_2) \cup \mathcal{C}(\ldots) \cup \mathcal{C}(G_k)$

Detecting Join Graphs

Important observation: the complement of a join graph is disconnected.

Detecting Join Graphs

Important observation: the complement of a join graph is disconnected.

Characteristics of join graphs

Let $G = G_1 + G_2$, then there is no edge in $\overline{G}$ with one vertex in $\overline{G_1}$ and another in $\overline{G_2}$. So $\overline{G_1}$ is either a connected component in $\overline{G}$ or a collection of connected component. If $\overline{G_1}$ has more than one connected component, G_1 is also a join graph. The same applies for G_2.

Detecting Join Graphs

Important observation: the complement of a join graph is disconnected.

Characteristics of join graphs

Let $G = G_1 + G_2$, then there is no edge in $\overline{G}$ with one vertex in $\overline{G_1}$ and another in $\overline{G_2}$. So $\overline{G_1}$ is either a connected component in $\overline{G}$ or a collection of connected component. If $\overline{G_1}$ has more than one connected component, G_1 is also a join graph. The same applies for G_2.

Therefore, a join graph G can be partitioned into the parts that were joined simply by finding the connected complements of $\overline{G}$ (using any traversal algorithm).

A Polynomial Time Solution for qC_5... ...and Some Problems

By partitioning the input graph and simply testing each part if it is a copy of C_5, the qC_5 family of join graphs can be solved in polynomial time.

A Polynomial Time Solution for qC_5... ...and Some Problems

By partitioning the input graph and simply testing each part if it is a copy of C_5, the qC_5 family of join graphs can be solved in polynomial time. However, a hard input can be generated by disguising a join graph, for instance, by adding an isolated vertex.

A Polynomial Time Solution for qC_5... ...and Some Problems

By partitioning the input graph and simply testing each part if it is a copy of C_5, the qC_5 family of join graphs can be solved in polynomial time.
However, a hard input can be generated by disguising a join graph, for instance, by adding an isolated vertex.
In general, a graph might not be a join graph, but contain subgraphs that are.

A More General Solution

Instead of trying to partition the graph only once, we can apply this idea to any branch and bound algorithm, by verifying if the subgraphs induced by the candidates set are join graphs and treating them separately.

New Algorithm

MaxClique($G, Q, K, C, test$)

```
1  if test = True and G[K] is a join graph then
2  |   for each part P of G[K] do
3  |   |   Q ← Q ∪ MaxClique(P, ∅, V(P), ∅, False) ; K ← K − V(P)
4  |   if |Q| > |C| then  C ← Q
5  else
6  |   pre–process the graph G and the sets Q and K
7  |   if K = ∅ then
8  |   |   if |Q| > |C| then  C ← Q
9  |   else
10 |   |   if |Q| + upper-bound(G, K) > |C| then
11 |   |   |   remove a vertex v from K
12 |   |   |   C ← MaxClique(G, Q ∪ {v}, K ∩ Γ_G(v), C, True)
13 |   |   |   C ← MaxClique(G, Q, K − {v}, C, False)
14 return C
```

Some Considerations on the New Algorithm

- Calls for MaxClique inside the test for join graphs are memoized.
- Parts are sorted by size in ascending order.
- Very small parts are treated directly:
  - size=1: return vertex
  - size=2: return any vertex
  - size=3: return edge if there is one, else return any vertex
- We based our implementation on one from Konc and Janežič (2007), available at http://insilab.org/maxclique/.

Experimental Results – DIMACS

	MCQD		New algorithm		Ratio (new/MCQD)	
Instance	**Tree size**	**Time (*s*)**	**Tree size**	**Time (*s*)**	**Tree size**	**Time**
MANN_a9	99	0.0002	69	0.0008	0.70	3.31
MANN_a27	48445	3.2034	47519	17.6710	0.98	5.52
brock200_1	235172	0.8043	201746	3.7553	0.86	4.67
brock200_2	3622	0.0099	3115	0.0419	0.86	4.23
brock200_3	12706	0.0470	10734	0.1810	0.84	3.85
brock200_4	46755	0.1346	45966	0.6890	0.98	5.12
brock400_1	118164924	546.4180	88017131	2420.7300	0.74	4.43
brock400_2	45097178	244.3200	54184600	1577.4200	1.20	6.46
brock400_3	113099318	479.5080	49363445	1292.6500	0.44	2.70
brock400_4	53805125	256.4530	14198686	487.9950	0.26	1.90
c-fat200-1	212	0.0005	8	0.0005	0.04	1.11
c-fat200-2	238	0.0006	12	0.0007	0.05	1.23
c-fat200-5	307	0.0013	30	0.0019	0.10	1.51
c-fat500-1	513	0.0021	11	0.0024	0.02	1.17
c-fat500-2	539	0.0025	24	0.0029	0.04	1.16
c-fat500-5	618	0.0047	41	0.0057	0.07	1.23
c-fat500-10	743	0.0112	2	0.0028	0.00	0.25

Experimental Results – DIMACS

	MCQD		New algorithm		Ratio (new/MCQD)	
Instance	**Tree size**	**Time (*s*)**	**Tree size**	**Time (*s*)**	**Tree size**	**Time**
hamming6-2	62	0.0002	9	0.0002	0.15	1.15
hamming6-4	123	0.0001	114	0.0004	0.93	3.03
hamming8-2	371	0.0088	20	0.0049	0.05	0.56
hamming8-4	12704	0.0543	11625	0.3118	0.92	5.75
hamming10-2	2863	2.5956	21673	68.5778	7.57	26.42
johnson8-2-4	50	0.0001	41	0.0002	0.82	2.93
johnson8-4-4	216	0.0006	138	0.0038	0.64	6.02
johnson16-2-4	583334	0.5403	525524	1.7433	0.90	3.23
keller4	7711	0.0198	8416	0.1095	1.09	5.51
p_hat300-1	2129	0.0074	1508	0.0212	0.71	2.87
p_hat300-2	7459	0.0307	4511	0.1670	0.60	5.45
p_hat300-3	629203	3.8099	440617	18.9078	0.70	4.96
p_hat500-1	10956	0.0290	8606	0.1259	0.79	4.34
p_hat500-2	193480	1.2548	108726	6.7789	0.56	5.40
p_hat500-3	25535382	276.9210	23115380	1858.4100	0.91	6.71
p_hat700-1	28084	0.0929	19245	0.3598	0.69	3.87
p_hat700-2	1091257	10.5074	701142	64.4854	0.64	6.14
p_hat1000-1	170075	0.4943	146247	2.0944	0.86	4.24
p_hat1000-2	31172704	359.9230	30089193	2936.3100	0.97	8.16
p_hat1500-1	1138690	3.9717	1046980	18.3531	0.92	4.62

Experimental Results – DIMACS

	MCQD		New algorithm		Ratio (new/MCQD)	
Instance	**Tree size**	**Time (*s*)**	**Tree size**	**Time (*s*)**	**Tree size**	**Time**
san200_0.7_1	1138	0.0058	69	0.0052	0.06	0.89
san200_0.7_2	2521	0.0086	1042	0.0199	0.41	2.32
san200_0.9_1	11031	0.0798	692	0.0464	0.06	0.58
san200_0.9_2	49351	0.4439	45070	2.4966	0.91	5.62
san200_0.9_3	332964	2.6274	1867762	74.1376	5.61	28.22
san400_0.5_1	2761	0.0102	437	0.0064	0.16	0.62
san400_0.7_1	42579	0.3171	27192	0.9350	0.64	2.95
san400_0.7_2	9182	0.1131	327177	7.5544	35.63	66.78
san400_0.7_3	510097	1.9279	957204	17.8615	1.88	9.26
san400_0.9_1	628188	19.9002	23986795	2313.5700	38.18	116.26
san1000	124031	0.4783	34919	0.6725	0.28	1.41
sanr200_0.7	106744	0.3240	84531	1.4133	0.79	4.36
sanr200_0.9	6361069	41.0665	4588339	191.7060	0.72	4.67
sanr400_0.5	242343	0.6782	216716	2.9644	0.89	4.37
sanr400_0.7	37811004	145.4990	33045120	670.5990	0.87	4.61

Experimental Results – qC_5 graphs

	MCQD		New algorithm		Ratio (new/MCQD)	
q	Tree size	Time (s)	Tree size	Time (s)	Tree size	Time
1	9	0.0003	7	0.0005	0.7778	1.6416
2	27	0.0007	8	0.0007	0.2963	0.9887
3	169	0.0042	13	0.0010	0.0769	0.2515
4	619	0.0197	18	0.0015	0.0291	0.0783
5	4773	0.1475	23	0.0021	0.0048	0.0145
6	19701	0.8146	28	0.0035	0.0014	0.0043
7	122827	5.2984	33	0.0044	0.0003	0.0008
8	752905	37.3874	38	0.0068	$5 \cdot 10^{-5}$	0.0002
9	3213301	206.3587	43	0.0065	$1 \cdot 10^{-5}$	$3 \cdot 10^{-5}$

References

Janez Konc and Dušanka Janežič. An improved branch and bound algorithm for the maximum clique problem. *MATCH Communications in Mathematical and in Computer Chemistry*, June 2007. URL `http://www.sicmm.org/konc/articles/match2007.pdf`.

Nikolay Lavnikevich. On the complexity of maximum clique algorithms: usage of coloring heuristics leads to the $\Omega(2^{n/5})$ algorithm running time lower bound, 2013. URL `http://arxiv.org/abs/1303.2546`.

Thank you!

Alexandre Prusch Züge
alexandrezuge@ufpr.br