INFORMATICA, 2016, Vol. 27, No. 2, 463–487

DOI: http://dx.doi.org/10.15388/Informatica.2016.95

Improved Infra-Chromatic Bound for Exact Maximum Clique Search

Pablo SAN SEGUNDO[1]*, Alexey NIKOLAEV[2], Mikhail BATSYN[2], Panos M. PARDALOS[2,3],

[1]*Center for Automation and Robotics and Polytechnic, University of Madrid C/Jose Gutiérrez Abascal, 2, 2006 Madrid, Spain*

[2]*Laboratory of Algorithms and Technologies for Network Analysis, National Research University Higher School of Economics, 136 Rodionova Street, Nizhny Novgorod, Russia*

[3]*Center for Applied Optimization, University of Florida 401 Weil Hall, P.O. Box 116595, Gainesville, FL 32611-6595, USA*

e-mail: pablo.sansegundo@upm.es

Received: December 2015; accepted: April 2016

Abstract. This paper improves an infra-chromatic bound which is used by the exact branch-and-bound maximum clique solver BBMCX (San Segundo *et al*., 2015) as an upper bound on the clique number for every subproblem. The infra-chromatic bound looks for triplets of colour subsets which cannot contain a 3-clique. As a result, it is tighter than the bound obtained by widely used approximate-colouring algorithms because it can be lower than the chromatic number. The reported results show that our algorithm with the new bound significantly outperforms the state-of-the-art algorithms in a number of structured and uniform random graphs.

Key words: maximum clique, approximate-colouring, branch-and-bound, combinatorial optimization, exact search, maximum satisfiability.

1. Introduction

Given a simple undirected graph $G = (V, E)$, a complete subgraph (or *clique*) is an induced subgraph such that all its vertices are pairwise adjacent. Determining whether a clique of a fixed size k exists, is a well known NP-complete problem referred to as *k-clique* (Karp, 1972). The *maximum clique problem* (MCP) is concerned with finding the largest possible clique in a graph, and the size of the solution is known as the clique number of the graph $\omega(G)$. Besides its theoretical relevance as an NP-hard problem, the MCP has many applications in different fields, such as bioinformatics (Konc and Janezic, 2010; Eblen *et al*., 2012; Butenko *et al*., 2009), coding theory,[1] network analysis,[2] computer vision (San Segundo and Artieda, 2015), robotics (San Segundo *et al*., 2010; San Segundo and Rodriguez-Losada, 2013) and others. In the rest of this section we

*Corresponding author.

[1]http://neilsloane.com/doc/graphs.html.

[2]http://www.networkrepository.com/.

present some definitions and a brief survey on efficient exact algorithms for the MCP related to this work.

1.1. *Definitions and Notation*

A simple undirected graph $G = (V, E)$ of order n consists of a set of vertices $V = \{1, 2, \ldots, n\}$ and a set of edges $E \subseteq V \times V$ that pair distinct vertices. A pair of vertices are said to be *adjacent* (or *neighbours*) if they are connected by an edge and $N(v)$ denotes the neighbour set of vertex v in G. Any subset of vertices $U \subseteq V$ *induces* a new subgraph $G[U] = (U, E[U])$ such that both endpoints of any edge in the new edge set $E[U]$ are in U.

A vertex *k-colouring* of G, $C(G)$, is an assignment of numbers (also colours or labels) from $\{1, 2, \ldots, k\}$ to every vertex of G, such that the endpoints of any edge take different colours. Any feasible *k-colouring* (or just *k-colouring* for simplicity) partitions the vertex set V in k disjoint subsets $C_1, C_2, \ldots, C_k$ such that $\bigcup_{i=1}^{n} C_i = V$ and each subset C_i is an independent set (a set of pairwise non-adjacent vertices). The number of different colours employed in a vertex colouring $|C(G)|$, is referred to as its *size*, and the size of a minimum colouring is also the chromatic number of the graph $\chi(G)$. An important property, which relates vertex colouring and maximum clique, is $\omega(G) \leqslant \chi(G)$ (Balas and Yu, 1986). It follows that the size of any vertex colouring is also an upper bound for $\omega(G)$.

We also consider the following definitions as necessary background in this work:

- *Degree of a vertex v* ($\deg(v)$): the number of vertices adjacent to v, alternatively $|N(v)|$;
- *Maximum graph degree* ($\Delta(G)$): the maximum degree of any of the vertices of the graph ($\max_{v \in V}\{\deg(v)\}$);
- *Maximal clique*: a clique that cannot be enlarged by any other vertex in the graph;
- *Greedy sequential vertex colouring* (SEQ): an approximate colouring procedure that iteratively assigns to a new vertex the smallest possible colour consistent with the numbers of previous vertices. SEQ requires a predefined vertex ordering and runs in $O(|V|^2)$;
- *Greedy-independent-set sequential vertex colouring* (ISEQ): an implementation of SEQ which computes colour sets (also called *colour classes*) sequentially. Consequently, it will not assign colour number k to any vertex unless all vertices with colour numbers below k have been numbered. Worth noting is that, for a given vertex ordering, ISEQ produces the same output as standard SEQ;
- *Width of a vertex* ($w(v)$): the number of preceding vertices adjacent to v. More formally, $w(v_i) = |N(v_i) \backslash \{v_{i+1}, v_{i+2}, \ldots, v_n\}|$;
- *Width of a vertex ordering*: the maximum width of any of its vertices;
- *Minimum-degree-last ordering of vertices*: a minimum-width ordering of vertices which results from iteratively *removing* vertices with minimum degree from V and placing them in *reverse order* in the new ordering.

1.2. *Branch-and-Bound Algorithms for the MCP*

In literature, there are many different approaches to the solving of the MCP, such as constraint programming, integer 0-1 programming, binary quadratic programming, copositive checking and programming (Žilinskas, 2011; Žilinskas and Dür, 2011) and others. However, the current more efficient exact maximum clique algorithms are branch-and-bound, that is, they combine maximal clique enumeration with pruning based on an upper bound on the clique number of every subproblem. In the last decade, greedy sequential vertex colouring SEQ has proved to be a very successful bound. In the comparison survey by Prosser (2012), MCS (Tomita *et al*., 2010) and the bit-parallel algorithm BBMC by San Segundo *et al*. (2011a, 2011b) were reported as the state-of-the-art solvers. Of the two, BBMC performed best over some well-known public data sets of small and middle-size dense graphs.

Better bounds on the clique number than the size of the SEQ colouring (but never below $\chi(G)$) may be obtained by a technique called *recolouring* (Re-NUMBER was the name given to the original procedure in MCS). Recolouring is a procedure which attempts to lower the colour number k of a vertex $v_1 \in C_k$ by a swap movement with two already coloured vertices ($v_2 \in C_i$, $v_3 \in C_j$) where $i < j < k$. BBMCR (San Segundo *et al*., 2011b) is the BBMC recolouring variant, and its bounding procedure runs in $O(|V|^3)$; BBMCR makes effective use of a bit string encoding for both ISEQ and recolouring.

Very recently, a way to reduce the maximum clique problem of a coloured graph to a partial maximum satisfiability problem (PMAX-SAT) has been described. The algorithms MaxCLQ (Li and Quan, 2010), and later IncMaxCLQ (Li *et al*., 2013), apply this reduction to each coloured subproblem during the MCP search, and use logical inferences to find k colour subsets which cannot possibly make part of a *k-clique* (referred to as *inconsistent subsets*). As a result, the upper bound k on the clique number for each inconsistent set is reduced by one, and the final bound for the whole subproblem can possibly fall below its chromatic number. Algorithm BBMCX (San Segundo *et al*., 2015) was the first time that the term *infra-chromatic* was used to describe a bound. Specifically, the infra-chromatic bound in BBMCX looks for groups of $k = 3$ inconsistent colour classes that do not contain triangles.

Also deserving attention (and very much related to this work) is *selective colouring*, another recent idea which was first described in BBMCL (San Segundo and Tapia, 2014). The intuition behind selective colouring is to relax SEQ to a partial colouring up to a certain pruning threshold. It has less overhead than computing the full colouring but prunes the search space somewhat less effectively. Branching on maximum colour has been taken up by the most effective algorithms since MCQ (Tomita and Seki, 2003), but in selective colouring all candidate vertices remain uncoloured – coloured vertices are all pruned – and selected according to their index, that is, the initial order fixed at the beginning of the search. In this paper we take the view of selective colouring and standard (full) colouring *as two different frameworks*, and all algorithms presented in this work are designed for one of the two, but not both.

This work contributes to the research on infra-chromatic bounds for the maximum clique problem in two ways. A first contribution is to describe BBMCX in the selective

colouring framework. A second contribution is a new bounding scheme which merges recolouring with the BBMCX infra-chromatic bound. The remaining contents of this work are divided as follows: Sections 2 and 3 cover prior related algorithms and the selective (colouring) framework. The new algorithms for exact maximum clique are described in Section 4 and validated empirically in Section 5. Finally, Section 6 presents conclusions and ongoing research.

2. Prior Related Algorithms

This section covers the necessary background required to understand contribution. It includes a state-of-the-art algorithm, which roughly corresponds to BBMC, and a brief explanation of the different pruning strategies: recolouring (Section 2.1), infra-chromatic bound (Section 2.2) and selective colouring (Section 2.3). Moreover, the amount of different recent improvements concerning this paper has led us to be particularly careful about algorithmic design. We consider the notion of an *algorithmic framework* as a family of algorithms related to a specific property. We also design procedures as templates so that different instantiations lead to different variants. Specifically, we classify exact maximum clique algorithms concerning this paper by the following two criteria:

- *The pruning framework*: *standard* – all vertices of a subproblem are assigned colour numbers through greedy-independent-set colouring, or *selective* – the selective colouring scheme;
- *The secondary pruning strategy*: recolouring, infra-chromatic or a combination of both.

From here onwards, the terms *standard* or *selective* will be used to refer to the two frameworks respectively.

As mentioned in the Introduction, most of the recent successful exact branch-and-bound maximum clique algorithms combine exhaustive enumeration of maximal cliques with a pruning strategy based on approximate vertex colouring. At each step of the search (typically a recursive call to the algorithm) a vertex selected for branching enlarges a clique, and each branch of the search tree contains a maximal clique. A solution to the problem is any clique in a leaf node with maximum cardinality. Before going into specific details, we list some useful definitions that which appear in the different algorithm listings described in this work:

- S: the clique to be enlarged at any point during search;
- $S_{\max}$: the incumbent (best) clique found at any point during search;
- U: the set of vertices of the current subproblem;
- U_v: the set of vertices of the child subproblem resulting from branching on vertex v;
- $L \subseteq U$: the set of candidate vertices for branching. L is sorted by colour in the standard framework, whereas in selective colouring vertices are sorted according to their index;
- $L_v \subseteq U_v$: the set of candidate vertices of the child subproblem which results from branching on vertex v;

Algorithm 1. A state-of-the-art exact maximum clique algorithm (standard framework).

Input: A simple graph $G = (V, E)$ sorted according to *minimum-degree-last*
Output: A maximum clique in $S_{\max}$
Initial values: $U \leftarrow V$, $S \leftarrow \emptyset$, $S_{\max} \leftarrow \emptyset$, $c(v_i) := \min\{i, \Delta(G) + 1\}\ \forall v_i \in V$, $L \leftarrow V$

REFCLQ($U, S, S_{\max}, C, L$)
1. **repeat until** $L = \emptyset$
2. select a vertex v from L in reverse order //maximum colour branching
3. **if** $(|S| + c(v) \leqslant |S_{\max}|)$ **then return** //pruning step
4. $U \leftarrow U \backslash \{v\}$
5. $S \leftarrow S \cup \{v\}$
6. $U_v \leftarrow U \cap N(v)$ //U_v preserves initial relative order of vertices
7. **if** $U_v \neq \emptyset$ **then** //checks for a maximal clique
8. $k_{\min} \leftarrow \max(|S_{\max}| - |S|, 1)]]$
9. $\{C^v, L_v\} \leftarrow$ UPPERBOUND($U_v, k_{\min}$)
10. REFCLQ($U_v, S, S_{\max}, C^v, L_v$)
11. **elseif** $|S| > |S_{\max}|$ **then** $S_{\max} \leftarrow S$
12. **endif**
13. $S \leftarrow S \backslash \{v\}$
14. **endrepeat**

- $c(v)$: the colour number assigned to vertex v;
- $k_{\min}$: a pruning (colour) threshold; vertices assigned a lower colour number will be pruned;
- C^v: the resulting colouring of the child subproblem U_v, ISEQ($G[U_v]$). Depending on the framework, it may be relaxed to the partial colouring $\{C_1, \ldots, C_{k_{\min}-1}\}$;
- F: a set of forbidden colour numbers employed in the infra-chromatic pruning scheme.

Algorithm 1 shows a pseudocode for REFCLQ, a state-of-the-art exact maximum clique algorithm representative of the standard framework. At each step (a recursive call to the algorithm) the clique determined by S is enlarged with a new vertex v from the set of candidate vertices L (step 5), until it becomes maximal. $S_{\max}$ stores the incumbent solution and is updated in step 11, if the maximal clique in S has greater cardinality. Pruning occurs in step 3 if the subproblem in the current step cannot improve $S_{\max}$, that is, $|S| + c(v) \leqslant |S_{\max}|$. This expression takes the form of $c(v) < k_{\min}$ in any derived child subproblem, for $k_{\min} = \max(|S_{\max}| - |S|, 1)$. The pruning step 3 is only valid combined with branching on maximum colour, an important idea first described in MCQ (Tomita and Seki, 2003) and taken up by most successful solvers since then. This is so because the algorithm reacts by backtracking – the *return* statement. Consider $v_j \in V$ to be the vertex that meets the pruning condition $|S| + c(v_j) \leqslant |S_{\max}|$, then not only is the induced subproblem $G[\{v_1, \ldots, v_j\}]$ pruned, but also subproblems $G[\{v_1, \ldots, v_{j-1}\}]$, $G[\{v_1, \ldots, v_{j-2}\}]$ and so on. In practice, branching on maximum colour is implemented by ordering the candidate vertices in L according to non-decreasing colour number, and selecting them in reverse order (see step 2).

Algorithm 2 describes the bounding procedure UPPERBOUND called by REFCLQ. Compared to similar procedures that may be found elsewhere, UPPERBOUND is the first

Algorithm 2. Pruning template UPPERBOUND called by REFCLQ.

Input: 1) An induced subproblem $G[U_v]$
2) A colour number threshold $k_{\min}$
Output: 1) A (partial) vertex colouring $C(G[U_v])$
2) A set L of candidate vertices sorted according to increasing colour number

UPPERBOUND($U_v, k_{\min}$)
Initial step: $U \leftarrow U_v$, $L \leftarrow \varphi$, $k \leftarrow 1$, $C \leftarrow \varphi$, $F \leftarrow \varphi$ //F is a set of forbidden colours

1. **repeat until** $U = \varphi$
2. $C_k \leftarrow U$
3. **repeat until** all vertices in C_k have been selected
4. choose the first vertex v from C_k not previously selected
5. result $\leftarrow$ FAIL
6. **if** $k \geqslant k_{\min}$ **then** //check if v can be pruned from branching list L
7. result $\leftarrow$ DO($v, k_{\min}, C_1, C_2, \ldots, C_{k_{\min}-1}, C_k, F$)
8. **endif** //$v \in C_k$, so its neighbours are removed from C_k
9. **if** *result* = FAIL **then** $C_k \leftarrow C_k \backslash N(v)$
10. $U \leftarrow U \backslash \{v\}$ //updates uncoloured vertices
11. **endrepeat**
12. $C \leftarrow C \cup C_k$
13. $k \leftarrow k + 1$
14. **endrepeat**
15. **for** $i = k_{\min}$ **to** $k - 1$
16. $L \leftarrow L \cup C_i$ //L is always sorted by colour number
17. **endfor**
18. **return** $(\{C_{k_{\min}}, C_{k_{\min}+1}, \ldots, C_{k-1}\}, L)$

example of the template algorithm design in this work. It calls a DO procedure (step 7) whose signature allows for different secondary pruning schemes (in this paper, recolouring, infra-chromatic or a combination of both). UPPERBOUND takes as input parameters the set of vertices of the child subproblem (to be bounded), and a colour number threshold $k_{\min}$. It outputs a (partial) colouring of the subproblem $C^v = C(G[U_v])$ and a set of candidate vertices L_v sorted according to non-decreasing colour number.

At the core of UPPERBOUND is greedy-independent-set colouring ISEQ implemented in two nested loops (steps 1 to 14). The inner loop (steps 3 to 11) is responsible for building each new colour set. The outer loop upkeeps the set of remaining uncoloured vertices, together with the number k of current colour set. Note that vertices with colour number below $k_{\min}$ are pruned and do not make part of the new candidate set L_v (steps 15 to 17).

Moreover, once the colour subset $C_{k_{\min}-1}$ is built, all remaining uncoloured vertices are passed to the concrete instantiation of DO in subsequent calls in an attempt to reduce their colour number k. In its most general form, DO receives as input the current colour class being built C_k, the vertex $v \in C_k$ object of study, the pruning threshold $k_{\min}$, the partial colouring up to the pruning threshold $\{C_1, C_2, \ldots, C_{k_{\min}-1}\}$ and a list of forbidden colours F. If v is reassigned a lower colour number than k in DO, both the partial colouration C_k and the set F are updated accordingly (the particular details will be explained for each specific instantiation of DO). Note that F is only used in the case of infra-chromatic bounds.

Algorithm 3. A procedure that implements *recolouring* in the standard framework. It instantiates DO in UPPERBOUND (Algorithm 2).

```
DO_RECOL(v, k_min, C_1, C_2, ..., C_{k_min-1}, C_k)                    //v ∈ C_k

 1. for k_1 := 1 to k_min − 2
 2.   if |C_{k_1} ∩ N(v)| = 1 then w ← C_{k_1} ∩ N(v)
 3.     for k_2 := k_1 + 1 to k_min − 1
 4.       if C_{k_2} ∩ N(w) = ∅ then
 5.         C_{k_1} ← C_{k_1} ∪ {v}                          //standard recolouring
 6.         C_{k_2} ← C_{k_2} ∪ {w}
 7.         C_k ← C_k\{v}
 8.         return SUCCESS
 9.       endif
10.     endfor
11.   elseif |C_{k_1} ∩ N(v)| = ∅ then
12.     C_{k_1} ← C_{k_1} ∪ {v}                              //one-move recolouring
13.     C_k ← C_k\{v}
14.     return SUCCESS
15.   endif
16. endfor
17. return FAIL
```

Regarding initial vertex ordering, it is well known that branching on vertices with minimum degree at the root node significantly reduces the size of the search tree in exact maximum clique search. In practice, vertices are ordered at the start of the search using a *minimum-degree-last* strategy and selected at the root node in *reverse order* (see Prosser, 2012; Matula and Beck, 1983; Carraghan and Pardalos, 1990, amongst others, for a detailed explanation).

Another important idea described both in MCS and BBMC (and denoted as an *implicit branching strategy*, San Segundo and Tapia, 2010), is to keep the initial vertex ordering fixed in all subproblems. This achieves a better compromise between overhead and quality of bounds than in earlier approaches (such as Tomita and Seki, 2003; Konc and Janezic, 2007). We note this is especially well suited for bit-parallel algorithms, like BBMC, because bitstring representations of vertex sets have a fixed order.

2.1. *Recolouring*

Recolouring is a repair mechanism for colour numbers that was first described in MCS (Tomita *et al*., 2010). Later, an improvement was proposed in San Segundo *et al*. (2011b) for the BBMC bit-parallel family of algorithms. Algorithm 3 outlines DO_RECOL, our algorithm for recolouring conceived as an instantiation of DO in UPPERBOUND (Algorithm 2). Note that signature F has been removed because it does not take part in recolouring and can be assumed empty without loss of generality.

The repair mechanism for SEQ described in DO_RECOL attempts to reassign input vertex $v \in C_k$, $k \geqslant k_{\min}$, a colour number lower than $k_{\min}$. It does so when there is a pair of colour sets (C_{k_1}, C_{k_2}) such that the following conditions hold:

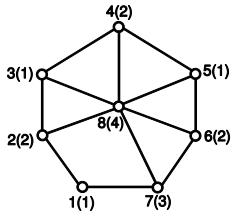

Fig. 1. A graph with coloured vertices.

I. $k_1 \leqslant k_{\min} - 2$ *and there is a single vertex* w *in* C_{k_1} *adjacent to* v;
II. $k_1 < k_2 \leqslant k_{\min} - 1$ *and no vertex in* C_{k_2} *is adjacent to* w.

If both colour sets (C_{k_1}, C_{k_2}) exist, a new colouring is obtained by assigning colour numbers k_1 and k_2 to vertices v and w respectively (steps 5 to 7). Noteworthy is that, because of the way SEQ works, colour number k_1 must strictly be lower than k_2 (see Proposition 1 in San Segundo *et al*., 2015). However, UPPERBOUND calls DO_RECOL for every candidate vertex, so it is perfectly possible that a colour subset C_{k_1} becomes *open* for v, that is, $C_{k_1} \cap N(v) = \emptyset$, after a prior successful recolouring. Step 11 in Algorithm 3 checks for that case. For additional details concerning this pruning scheme we refer the reader to Tomita *et al*. (2010), San Segundo *et al*. (2011b).

2.2. *Infra-Chromatic Upper Bound for Maximum Clique*

Let us consider the simple graph G, with chromatic number $\chi(G) = 4$, depicted in Fig. 1, in which vertex {8} has been assigned colour number 4 by SEQ (colours are shown in parenthesis for each vertex). Thus, 4 is the upper bound for the clique number and vertex {8} can at most belong to a clique of size 4, in a branching-on-maximum-colour strategy. Moreover, since 4 is also the size of the colouring, it also bounds the size of any clique hidden in the graph.

The *infra-chromatic bound* described in San Segundo *et al*. (2015) is based on the following observation: *any three vertices of a clique always belong to a different colour class in any colouring, and always form a 3-clique*. If, for vertex {8}, we can find two colour subsets with colour numbers less than 4 in which there are no vertices which form a 3-clique with vertex 8, then vertex 8 cannot belong to a clique of size 4 and thus the infra-chromatic bound for it is 3. In the example, two such colour classes are the third colour set $C_3 = \{7\}$ and the first colour set $C_1 = \{1, 3, 5\}$. The three sets $C_4 = \{8\}$, C_1 and C_3 cannot form a 3-clique and are referred to as *inconsistent sets*, or an inconsistent triplet of sets.

The following proposition explains the infra-chromatic bound for the general case:

Proposition. *If vertex* v *has colour* k *in a graph coloured with* k *colours and there exist two colour classes with colour numbers below* k *in which there are no vertices forming a* 3*-clique with vertex* v*, then vertex* v *cannot belong to a clique of size* k *or bigger.*

Algorithm 4. The reference algorithm for selective colouring.

Input: A graph $G = (V, E)$ with vertices sorted according to *minimum-degree-last*
Output: A maximum clique in vertex set $S_{\max}$
Initial values: $U \leftarrow V$, $S \leftarrow \emptyset$, $S_{\max} \leftarrow \emptyset$, $L \leftarrow V$

REFCLQ$_S(U, S, S_{\max}, L)$
1. **repeat until** $L = \varphi$
2. select a vertex v from L in reverse order
3. $U \leftarrow U \backslash \{v\}$
5. $S \leftarrow S \cup \{v\}$
6. $U_v \leftarrow U \cap N(v)$
7. **if** $U_v \neq \emptyset$ **then**
8. $k_{\min} \leftarrow \max(|S_{\max}| - |S|, 1)$
9. $L_v \leftarrow$ UPPERBOUND$_S(U_v, k_{\min})$
10. REFCLQ$_S(U_v, S, S_{\max}, L_v)$
11. **elseif** $|S| > |S_{\max}|$ **then** $S_{\max} \leftarrow S$
12. **endif**
13. $S \leftarrow S \backslash \{v\}$
14. **endrepeat**

Proof. Vertex v cannot belong to a clique of size bigger than k because otherwise the graph cannot be coloured in k colours. Assume that vertex v belongs to a clique of size k. Since all vertices of this clique have to be coloured in different colours, as all are pairwise adjacent, then each colour class from 1 to k will have exactly one vertex from this clique. So any two colour classes will contain two vertices which form a 3-clique with vertex v. This contradicts the conditions of the proposition, so our assumption is wrong and vertex v cannot belong to a clique of size k. □

With the help of this proposition it is easy to reduce by one the colour bound $k = 4$ for the graph in Fig. 1. We call this reduced bound *infra-chromatic*, and the vertex $v = \{8\}$, together with the two colour sets (C_1, C_3), constitute the inconsistent triplet. In practice, to reduce the overhead when searching for two such colour sets, we require that one of them has only one vertex adjacent to vertex v (as in the example). In BBMCX (San Segundo *et al.*, 2015), an instantiation of DO – DO_INFRA-CHROMATIC, Algorithm 4 – is presented for such infra-chromatic bound. Reported results showed good pruning ability and improved efficiency for a number of graphs from public data sets. A first contribution of this paper is to enhance BBMCX with selective colouring. This is equivalent to adapt it to our proposed selective framework.

2.3. *Selective Colouring*

Selective colouring is a recent idea described in BBMCL (San Segundo and Tapia, 2014) in which only a partial SEQ colouring of size $k_{\min} - 1$ is computed for every subproblem. The coloured vertices are those in the child subproblem that cannot improve the incumbent solution in $S_{\max}$. In its default version, the algorithm no longer branches by maximum colour – it is not possible now, since candidate vertices remain uncoloured – but selects

Algorithm 5. Outline of the pruning template in selective colouring.

Input: An induced subgraph $G[U_v]$ and a colour number threshold $k_{\min}$
Output: A set of candidate vertices L

UPPERBOUND$_S(U_v, k_{\min})$
Initial step: $U \leftarrow U_v$, $L \leftarrow \emptyset$, $k \leftarrow 1$

1. **repeat until** $k = k_{\min}$
2. $C_k \leftarrow U$
3. **repeat until** all vertices in C_k have been selected
4. choose the first vertex v from C_k not previously selected
5. $C_k \leftarrow C_k \backslash N(v)$
6. $U \leftarrow U \backslash \{v\}$
7. **endrepeat**
8. $C \leftarrow C \cup C_k$
9. $k \leftarrow k + 1$
10. **endrepeat**
11. $L \leftarrow U$ //stores vertices v such that $c(v) \geqslant k_{\min}$
12. DO$_S(L, C = \{C_1, C_2, \ldots, C_{k_{\min}-1}\})$ //additional pruning (recolouring, infra-chromatic, etc.)
13. **return** L

vertices in reverse order of their index instead. Consequently, selective colouring is less informed than standard (full) colouring, but has less overhead: no explicit pruning step, no explicit storage of colour numbers and less colours to compute.

The next section describes the selective colouring framework. We continue with the same design of prior algorithms, and use a template function to represent an abstract secondary pruning scheme which can either be infra-chromatic, recolouring or a combination of both. The subscript S (for Selective) will be added to the names of procedures to disambiguate from the standard framework.

3. The Selective Colouring Framework

We conceive the selective colouring framework as the family REFCLQ$_S$ of algorithms outlined in Algorithm 4. The description is similar to Algorithm 1, but now pruning is reduced to the call to UPPERBOUND$_S$ and there is no initial colouring at the root node, no explicit pruning step and no colouring information returned by UPPERBOUND$_S$.

Algorithm 5 presents a pseudocode for UPPERBOUND$_S$. It prunes the search space by storing in L, ordered by index, only those vertices from U which *do not* belong to the partial colouring $\{C_1, C_2, \ldots, C_{k_{\min}-1}\}$. The latter is computed in two nested loops (steps 1 to 10).

In step 12, UPPERBOUND$_S$ calls template procedure DO$_S$ which represents the secondary pruning scheme. Note that its signature is simpler than in the standard framework, that is, just the partial colouring $\{C_1, C_2, \ldots, C_{k_{\min}-1}\}$ and the branching set of vertices L. If DO$_S$ succeeds, it will remove the pruned vertices from L. We also note that DO$_S$ is called *after* the partial colouring is computed, which was not the case in DO.

Algorithm 6. Outline of the infra-chromatic pruning procedure in the selective colouring framework.

```
DO_ITER(L, C = {C_1, C_2, ..., C_{k_min-1}})
Variables: A set of forbidden colour numbers F (initially F ← ∅)

1.  repeat until all vertices in L have been examined
2.  select a new vertex v from L in order
3.     result ← FILTER(v, C, F)
4.     if result = SUCCESS then
5.           L ← L\{v}                                  //filters vertex
6.     endif
7. endrepeat
```

This paper contributes to the infra-chromatic bound with different instantiations of DO_S: infra-chromatic alone and the combination recolouring + infra-chromatic. They are described in the next section.

4. Infra-Chromatic Bounds in the Selective Colouring Framework

The infra-chromatic procedure, described in Section 2.2, can now be integrated into the selective colouring framework because of the careful design of REFCLQ_S (Algorithms 4 and 5). Specifically, we implement DO_S in $\mathrm{UPPERBOUND}_S$ with the help of two new procedures: DO_ITER and FILTER (Algorithm 6).

DO_ITER is the actual instantiation of DO_S, and is conceived as an iterator over all vertices in L. It also initializes the list F of forbidden colours to the empty set and calls FILTER (the real pruning procedure) for every vertex in L (step 3). FILTER is also designed as a template function to capture different infra-chromatic schemes. Its signature contains the vertex v considered for pruning, the current partial colouring $\{C_1, C_2, \ldots, C_{k_{\min}-1}\}$ of the selective framework and a list F of forbidden colour numbers. Depending on the particular bounding scheme, F is updated as follows:

- *Recolouring*: here F has no impact and may be considered the empty set. A recolouring procedure, which may be instantiated directly in the proposed selective framework because it has the same signature (excluding F), may be found in listing 7 of BBMCL (Balas and Yu, 1986). We also refer the reader to DO_RECOL in Algorithm 3 for the description of recolouring in the standard framework.
- *Infra-chromatic*: once a colour subset has been proven inconsistent, it cannot take part in further inferences, so its colour number is stored in F and filtered out from other candidate inconsistent triplets in successive calls to DO_ITER.

If the pruning attempt succeeds, FILTER returns SUCCESS and signals DO_ITER to remove the pruned vertex from L in step 5. This process continues until all candidate vertices have been examined.

Algorithm 7. The infra-chromatic filter for selective colouring.

```
FILTER_INFRACHROM_S(v, C_1, C_2, ..., C_{k_min-1}, F)        //F is the list of forbidden colours
 1.  for k_1 := 1 to k_min - 1
 2.    if k_1 ∈ F then continue
 3.    if |C_{k_1} ∩ N(v)| = 1 then w ← C_{k_1} ∩ N(v)
 4.        for k_2 := 1, ..., k_1 - 1, k_1 + 1, ..., k_min - 1
 5.            if k_2 ∈ F then continue
 6.            if C_{k_2} ∩ N(w) ∩ N(v) = ∅ then          //inconsistent triplet condition
 7.                F ← F ∪ {k_1} ∪ {k_2}                  //tags the colours as forbidden
 8.                return SUCCESS
 9.            endif
10.        endfor
11.    endif
12. endfor
13. return FAIL
```

4.1. *The Basic Infra-Chromatic Bound*

FILTER_INFRACHROM$_S$ (Algorithm 7) describes how to instantiate FILTER for infra-chromatic pruning in the selective framework. Compared with procedure DO_INFRACHROMATIC employed by BBMCX (listing 4 of San Segundo *et al*., 2015), in the selective colouring framework the control flow changes because of DO_ITER. A brief explanation now follows, necessary to understand the new combined *recolouring + infra-chromatic* filter described in the next section.

FILTER_INFRACHROM$_S$ searches for inconsistent triplets of the form $(C_{k_1}, C_{k_2}, \{v\})$, in which the last colour subset is a singleton containing the input vertex attempted to be pruned. According to the explanation in Section 2.2, an inconsistent triplet requires that the following two conditions are met:

$$\begin{cases} \text{I.} & C_{k_1} \cap N(v) = w \\ \text{II.} & C_{k_2} \cap N(w) \cap N(v) = \emptyset \end{cases}$$

Condition I (implemented in step 3) indicates that vertex w is the *only* adjacent vertex to v in the first colour subset C_{k_1}. We require this set to be a singleton in order to reduce computational time to find inconsistent triplets. Condition II (implemented in step 6) establishes that there is no vertex adjacent to both v and w in the second colour subset C_{k_2}. This implies that vertices v, w and any other vertex from C_{k_2} cannot form a 3-clique, and therefore the triplet $(C_{k_1}, C_{k_2}, \{v\})$ is inconsistent. It is worth mentioning here that condition I is also required for recolouring. This is relevant to implement the efficient combined *recolouring + infra-chromatic* filter, as will be shown in the next subsection.

Once an inconsistent triplet has been detected, colour numbers k_1 and k_2 are stored in F (step 7) and steps 2 and 5 ensure that they will not make part of future inconsistent triplets in later calls. It is important to take into account that pruning conditions I and II are *not* symmetrical with respect to the colour pair (C_{k_1}, C_{k_2}), that is, it is possible that the triplet $(C_{k_1}, C_{k_2}, \{v\})$ meets both conditions but $(C_{k_2}, C_{k_1}, \{v\})$ does not, or vice

Algorithm 8. The new *recolouring + infra-chromatic* pruning procedure for selective colouring.

```
FILTER_RECOL_INFRACHROM_S(v, C_1, C_2, ..., C_{k_min-1}, F)
 1.  for k_1 := 1 to k_min - 1
 2.    if k_1 ∈ F then continue
 3.    if |C_{k_1} ∩ N(v)| = 1 then w ← C_{k_1} ∩ N(v)
 4.        for k_2 := 1, ..., k_1 - 1, k_1 + 1, ..., k_min - 1
 5.            if k_2 ∈ F then continue
 6.            if C_{k_2} ∩ N(w) ≠ ∅ then
 7.                if C_{k_2} ∩ N(w) ∩ N(v) = ∅ then
 8.                    F ← F ∪ {k_1} ∪ {k_2}          //infra-chromatic pruning
 9.                    return SUCCESS
10.                endif
11.            else                                   //standard recolouring
12.                C_{k_1} ← C_{k_1} ∪ {v}\{w}
13.                C_{k_2} ← C_{k_2} ∪ {w}
14.                return SUCCESS
15.            endif
16.        endfor
17.    elseif C_{k_1} ∩ N(v) = φ then
18.        C_{k_1} ← C_{k_1} ∪ {v}                    //one-move recolouring
19.        return SUCCESS
20.    endif
21. endfor
22. return FAIL
```

versa. Also worth noting is that, compared with the standard framework, the signature of FILTER_INFRACHROM$_S$ *does not* include the colour subset to which vertex v was assigned prior to the call (since now vertex v has not been coloured) nor $k_{\min}$ (implicit in the partial colouring of the signature).

4.2. *A Combined Recolouring and Infra-Chromatic Bound*

As pointed out in connection with FILTER_INFRACHROM$_S$ in Algorithm 7, recolouring is related to our proposed infra-chromatic filter by condition I. In both cases, a colour subset C_{k_1} is required to have only one vertex $w \in C_{k_1}$ belonging to the neighbourhood of v. Consequently, it is possible to define an efficient combined *recolouring + infra-chromatic* filter that exploits this circumstance.

FILTER_RECOL_INFRACHROM$_S$, in Algorithm 8, outlines one such combined procedure. Its control flow, two nested loops to enumerate colour subset pairs, is now combined with the possibility of recolouring. In previous FILTER_INFRACHROM$_S$, once an inconsistent triplet has been found, the colours are tagged as forbidden in F and removed from further inferences. It is therefore a good strategy to *enlarge* colour subsets as much as possible before they make part of an inconsistent triplet. FILTER_RECOL_INFRACHROM$_S$ follows this strategy: after condition I has been met in step 3, vertex w is first checked for a possible colour reassignment to one of the colour sets not yet tagged as forbidden (step 6). It is easy to see that the recolouring

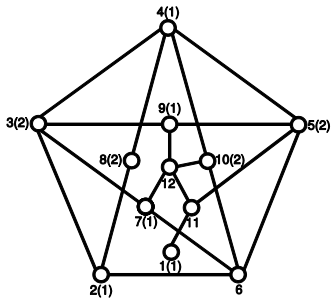

Fig. 2. A subgraph for algorithm demonstration.

condition $C_{k_2} \cap N(w) = \emptyset$ is stronger than the one required for the infra-chromatic filter $C_{k_2} \cap N(w) \cap N(v) = \emptyset$, *so recolouring should be attempted before* condition II. Step 6 discriminates between recolouring and infra-chromatic pruning. If the condition $C_{k_2} \cap N(w) \neq \emptyset$ holds, then recolouring is no longer possible and infra-chromatic pruning is attempted in step 7. If the latter is successful, step 8 tags the colour numbers of the corresponding inconsistent colour subsets as forbidden. If, on the other hand, colour subset C_{k_2} is open for vertex w according to step 6, then recolouring takes place and colour sets C_{k_1}, C_{k_2} are updated appropriately (steps 12 to 13).

Finally, we note that colour subsets may become open for a particular vertex after successive calls to FILTER_RECOL_INFRACHROM$_S$, in a similar fashion as in normal recolouring. This was not possible in FILTER_INFRACHROM$_S$ and is now checked in step 17.

4.3. *An Example for Algorithm Demonstration*

We demonstrate the work of the algorithm on the graph shown in Fig. 2. We assume it depicts the subgraph $G[U_v]$ of a vertex v which has just been selected in step 2 of REFCLQ$_S$ (Algorithm 4). We also assume that the incumbent (largest) clique found so far has size 3 and that $|S| = 0$. Consequently, $U_v = \{1, \ldots, 12\}$, $|S_{\max}| = 3$ and $k_{\min} = 3$. We now call UPPERBOUND$_S$ (Algorithm 5) with these parameters. The main calculations for this example are shown below. The colouring obtained by UPPERBOUND$_S$ is shown in Fig. 2 in brackets.

4.4. *Discussion on the Different Algorithms*

The previous subsections have shown how to extend infra-chromatic bounds to the selective framework. Moreover, a new combined recolouring + infra-chromatic pruning procedure FILTER_RECOL_INFRACHROM$_S$ has been proposed for selective colouring. To summarize, Fig. 3 depicts the different algorithms which have been mentioned in the paper.

UPPERBOUND$_S$ call for the subproblem $G[U_v]$ shown in Fig. 2.
$C_1 = \{1, \ldots, 12\}$
$v = 1$, $C_1 = \{1, \ldots, 10, 12\}$, $U = \{2, \ldots, 12\}$
$v = 2$, $C_1 = \{1, 2, 4, 5, 7, 9, 10, 12\}$, $U = \{3, \ldots, 12\}$
$v = 4$, $C_1 = \{1, 2, 4, 7, 9, 12\}$, $U = \{3, 5, \ldots, 12\}$
$v = 7$, $C_1 = \{1, 2, 4, 7, 9\}$, $U = \{3, 5, 6, 8, \ldots, 12\}$
$v = 9$, $C_1 = \{1, 2, 4, 7, 9\}$, $U = \{3, 5, 6, 8, 10, 11, 12\}$
$C_2 = \{3, 5, 6, 8, 11, 12\}$
$v = 3$, $C_2 = \{3, 5, 6, 8, 10, 11, 12\}$, $U = \{5, 6, 8, 10, 11, 12\}$
$v = 5$, $C_2 = \{3, 5, 8, 10, 12\}$, $U = \{6, 8, 10, 11, 12\}$
$v = 8$, $C_2 = \{3, 5, 8, 10, 12\}$, $U = \{6, 10, 11, 12\}$
$v = 10$, $C_2 = \{3, 5, 8, 10\}$, $U = \{6, 11, 12\}$
We stop here because $k_{\min} = 3$. So we have two colour subsets $C_1 = \{1, 2, 4, 7, 9\}$ and $C_2 = \{3, 5, 8, 10\}$. Moreover, according to the selective colouring framework the set of uncoloured vertices $U = \{6, 11, 12\}$ is also the candidate set L of vertices for branching.

DO_ITER($L, C = \{C_1, C_2\}$)
$v = 6$, FILTER_RECOL_INFRACHROM$_S(v, C_1, C_2, F)$
$k_1 = 1$, $C_{k_1} \cap N(v) = \{2, 7\}$
$k_1 = 2$, $C_{k_1} \cap N(v) = \{5, 10\}$
return FAIL
$v = 11$, FILTER_RECOL_INFRACHROM$_S(v, C_1, C_2, F)$
$k_1 = 1$, $C_{k_1} \cap N(v) = \{1\}$, $w = 1$
$k_2 = 2$, $C_{k_2} \cap N(w) = \emptyset$ // standard recolouring
$C_{k_1} = \{1, 2, 4, 7, 9\} \cup \{11\} \backslash \{1\} = \{2, 4, 7, 9, 11\}$
$C_{k_2} = \{3, 5, 8, 10\} \cup \{1\} = \{1, 3, 5, 8, 10\}$
return SUCCESS
$L = \{6, 11, 12\} \backslash \{11\} = \{6, 12\}$ //filters vertex
$v = 12$, FILTER_RECOL_INFRACHROM$_S(v, C_1, C_2, F)$
$k_1 = 1$, $C_{k_1} \cap N(v) = \{7, 9, 11\}$
$k_1 = 2$, $C_{k_1} \cap N(v) = \{10\}$, $w = 10$
$k_2 = 1$, $C_{k_2} \cap N(w) = \{4\}$
$C_{k_2} \cap N(w) \cap N(v) = \emptyset$ //infra-chromatic pruning
$F = \{1, 2\}$
return SUCCESS
$L = \{6, 12\} \backslash \{12\} = \{6\}$ //filters vertex

Before discussing performance, a brief comment on algorithm design. FILTER_ECOL_INFRACHROM$_S$ (Algorithm 8) in its present form may only be used in the context of selective colouring, that is, called by REFCLQ$_S$ in Algorithm 4. It may nevertheless be adapted to the standard framework, that is, called from REFCLQ in Algorithm 1, with the following minor changes:

– *Remove DO_ITER procedure* (Algorithm 6): FILTER_RECOL_INFRACHROM should map to template DO in UPPERBOUND (Algorithm 2). Consistency with signatures indicates that $k_{\min}$ and C_k (the colour class containing the vertex to be analysed) should be passed to any instantiation.

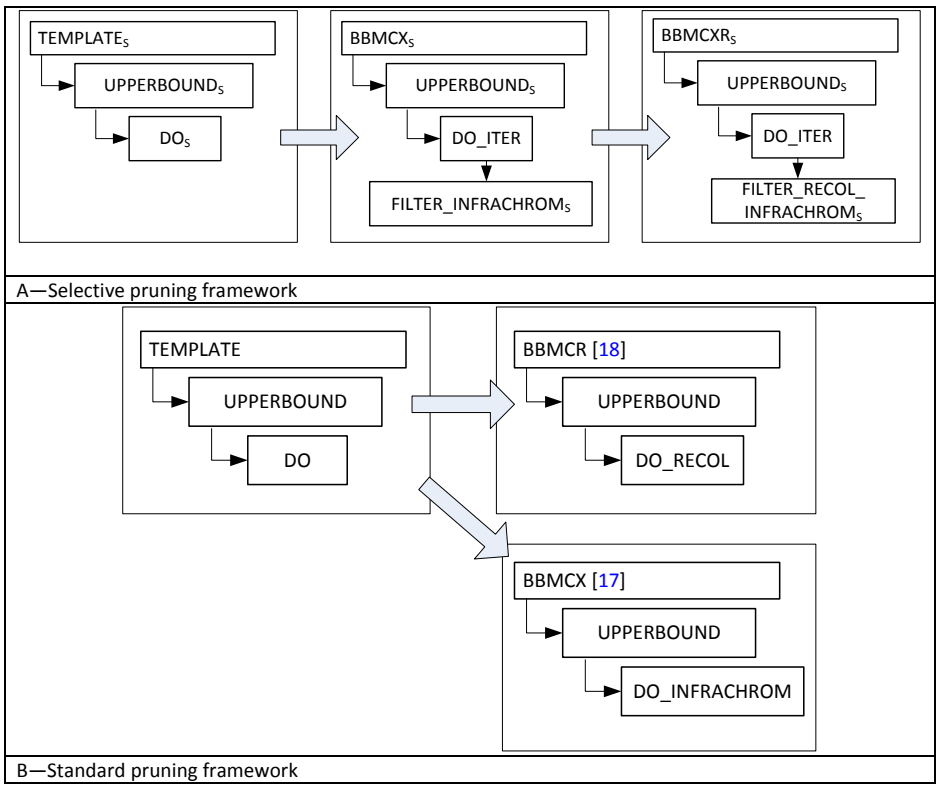

Fig. 3. Summary of the different algorithmic variants related to this work. The new algorithms described in the paper correspond with the selective framework (A).

– *Update C_k parameter appropriately when the pruning succeeds*: in particular, vertex v should be removed from C_k just before steps 9, 14 and 19 of FILTER_RECOL_INFRACHROM$_S$.

This example shows that switching between frameworks is possible with little effort, and we believe it validates the template design employed in this work.

We now present some general considerations in relation with the different frameworks and secondary level pruning schemes considered. The standard framework is expected to produce, on average, smaller search trees since it branches on vertices with maximum colour number, whereas selective colouring is less informed. On the other hand, the former has greater overhead. This is extensible to the infra-chromatic scheme: smaller search trees are to be expected in algorithms which employ it, but again with additional computational cost. The combined *recolouring* + *infra-chromatic* pruning scheme introduces even greater overhead, but both filters paste well together due to condition I. This new filter is therefore expected to produce the smallest search trees, and perform well in the harder and denser graphs.

To end this discussion, we draw attention to the fact that greedy-independent-set colouring ISEQ has been applied in both frameworks (specifically in UPPERBOUND and

UPPERBOUND$_S$). While this is not critical for recolouring, it *is* for the infra-chromatic scheme which requires that the $k_{\min} - 1$ colour sets, starting point of the bounding procedure, are *not* enlarged by new vertices as colouring proceeds. Tagging a colour subset C_k as forbidden, and later having SEQ assign a colour number k to an unfiltered vertex, results in incompleteness. While other alternatives to ISEQ are possible, it is a simple strategy which guarantees that this will not occur. Moreover, ISEQ has been found to be a good choice for bit-parallel exact maximum clique algorithms for other reasons as well (San Segundo *et al*., 2011a).

5. Experiments

To validate the new infra-chromatic bound, and specifically the FILTER_RECOL_INFRACHROM$_S$ filter, we have considered the following algorithms:

(a) BBMCX (San Segundo *et al*., 2015): the prior algorithm described in the standard framework which employs an infra-chromatic bound.
(b) BBMCXRS: the combined *recolouring* + *infra-chromatic* pruning scheme described in Algorithm 8.
(c) IncMaxCLQ (Li *et al*., 2013): the most recent PMAX-SAT-based algorithm, which can also compute maximum clique bounds for subproblems below their chromatic number.

The hardware used in the experiments was a 20 core XEON with 64 GB of RAM running a Linux OS. All algorithms considered were run on a single core of this machine. To allow other researchers to compare with our hardware, we ran the DIMACS benchmark program *dfmax* in a standard fashion (Johnson and Trick, 1996). The running times obtained over DIMACS machine benchmark graphs $r300.5$, $r400.5$ and $r500.5$ were 0.167, 0.900 and 3.430 seconds, respectively.

We refer the reader to the BITSCAN and GRAPH libraries (BITSCAN C++ library, GRAPH C++ library) for implementation details concerning the bitstring encoding of the BBMC family of algorithms. The source code for prior BBMC variants, i.e. BBMC, BBMCR, BBMCL, is available in BBMC releases (2016), and so are Win32 64-bit binaries for the new algorithms described in this work.[3] Finally, we note that IncMaxCLQ was provided to us by its main developer Chu-Min Li and run with default parameters.

BBMCXR$_S$ also incorporates the latest enhancements in preprocessing. Regarding initial sorting of vertices, it uses the same procedure described for IncMaxCLQ, and selects the *best* between the standard *mininum-degree-last* sorting, as in Algorithm 1, and a colour-based ordering. The initial sorting procedure is similar to the one described in Li *et al*. (2013) with minor changes: first, a greedy colouring is obtained using the constructive recursive-largest-first (RLF) colouring heuristic (Leighton, 1979). If the highest colour classes of the colouring contain more than one colour class with a single vertex, RLF is dismissed and standard *mininum-degree-last* sorting is applied. If this is not the

[3] http://venus.elai.upm.es/logs/results_infrachrom/.

case, vertices are arranged according to the colour labels given by RLF in increasing order. We also compute a *good* initial solution by running leading approximate heuristic ILS, as recommended in Maslov *et al*. (2013).

With respect to the public data sets used for validation, we provide performance over 67 structured graphs and a number of uniform random graphs which may be found in other maximum clique reports elsewhere. The majority of structured instances were presented at the Second DIMACS Implementation Challenge[4] (Johnson and Trick, 1996). The rest are taken from the BHOSHLIB benchmark[5] (specifically, the *frb30–15* collection). Concerning uniform random graphs $G_{n,p}$ – where n is the number of vertices, and p the probability that there exists an edge between any two vertices – reported times for every (n, p) pair are averaged over 10 runs.

5.1. *Results*

Table 1 reports performance of the three algorithms considered over structured graphs. Header column NEW refers to BBMCXR$_S$. The last two columns show a time ratio of prior BBMCX and IncMaxCLQ to BBMCXR$_S$. Cells in bold report the best time for each graph with precision of milliseconds.

Out of the 67 structured graphs reported, BBMCXR$_S$ performs better than IncMaxCLQ in 60 of them, and better than BBMCX in 50. Moreover, it shows best times overall in 54 graphs. IncMaxCLQ performs significantly better than BBMCXR in the *keller5* graph, in the two graphs of the *C* family and in *sanr200_0.9*. In these 4 cases, the extra effort spent to compute better bounds prunes the search space efficiently. As to BBMCX, it is only significantly faster than the new algorithm in very easy instances.

Experiments with uniform random graphs are reported in Table 2. In contrast with the previous case, here the differences between BBMCX and the new BBMCXR$_S$ algorithm are much less acute, possibly because the more sophisticated bound of BBMCXR$_S$ prunes better when there is structure to capture. Still, BBMCXR$_S$ performs best in 19 cases out of the 27 graphs considered, and is more than 5 times faster than IncMaxCLQ in 5 graphs. We note that IncMaxCLQ outperforms new BBMCXR$_S$ only in the very dense graphs ($p \geqslant 0.9$).

6. Conclusions and Future Work

The encoding of maximum clique to a partial maximum satisfiability problem has opened up an important line of research in exact maximum clique algorithms. One of the latest ideas is to implement infra-chromatic pruning as described in the recently published bit-parallel algorithm BBMCX.

This paper presents several improvements over BBMCX. We first describe a new algorithm BBMCX$_S$, which enhances prior BBMCX with *selective colouring*, a recent

[4] http://cs.hbg.psu.edu/txn131/clique.html.

[5] http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm.

Table 1
Performance of the algorithms considered over structured instances. In bold, best times for each row. A *step* is a recursive call to the algorithm. The columns with header p show average density, and header ω is the size of the solution. The time limit was fixed at 3h and is measured in seconds with millisecond precision.

			IncMaxCLQ (Li *et al.*, 2013)		BBMCX (San Segundo *et al.*, 2015)		NEW		CLQ/NEW	BBMCX/NEW
	p	ω	Steps	Time	Steps	Time	Steps	Time	Time	Time
C125.9	0.90	34	48	**0.017**	2153	**0.017**	1891	0.026	0.7	0.7
C250.9	0.90	44	734186	**228**	74379132	912	50578323	638	0.4	1.4
MANN_a27	0.99	126	290	0.266	4679	**0.236**	4521	0.255	1.0	0.9
MANN_a45	0.99	345	21595	101	118384	**39.8**	98801	66.9	1.5	0.6
MANN_a9	0.93	16	0	0.002	16	**< 0.001**	14	**< 0.001**	1.5	1.0
brock200_1	0.75	21	2700	0.482	33445	0.241	28436	**0.190**	2.5	1.3
brock200_2	0.50	12	112	0.016	277	**0.002**	151	0.004	4.1	0.5
brock200_3	0.61	15	493	0.046	1519	0.019	1523	**0.010**	4.6	1.9
brock200_4	0.66	17	434	0.057	6358	0.040	2816	**0.028**	2.0	1.4
brock400_1	0.75	27	890529	182	17608242	195	13955624	**149**	1.2	1.3
brock400_2	0.75	29	711295	139	6929601	81.3	4053581	**51.9**	2.7	1.6
brock400_3	0.75	31	887764	170	14139535	145	891968	**16.0**	11	9.1
brock400_4	0.75	33	640279	129	6001694	67.8	365855	**7.90**	16	8.6
brock800_1	0.65	23	39290054	7846	159471348	2674	153555219	**2419**	3.2	1.1
brock800_2	0.65	24	49803966	10011	142768093	2406	123868106	**2038**	4.9	1.2
brock800_3	0.65	25	17522630	3759	91142593	1567	59703387	**1120**	3.4	1.4
brock800_4	0.65	26	24600360	4768	59354622	1092	52398010	**937**	5.1	1.2
dsjc1000.1	0.10	6	81	0.094	299	**0.003**	265	0.004	23	0.8
dsjc1000.5	0.50	15	1562003	197	6516009	95.7	5988617	**82.2**	2.4	1.2
dsjc500.1	0.10	5	0	0.021	18	**< 0.001**	13	**< 0.001**	21	1.0
dsjc500.5	0.50	13	19303	2.63	122680	0.936	107408	**0.733**	3.6	1.3
frb30-15-1	0.82	30	0	0.132	58372163	702	1	**< 0.001**	132	701928
frb30-15-2	0.82	30	8	0.125	47285091	562	1	**< 0.001**	125	561604
frb30-15-3	0.82	30	23	0.171	45231516	512	1	**< 0.001**	171	511694
frb30-15-4	0.82	30	22	0.213	136241566	1492	1	**< 0.001**	213	1491879
frb30-15-5	0.82	30	9	0.163	52198654	570	1	**< 0.001**	163	569641

(*continued on next page*)

Table 1
(*continued*)

			IncMaxCLQ (Li *et al.*, 2013)		BBMCX (San Segundo *et al.*, 2015)		NEW		CLQ/NEW	BBMCX/NEW
	p	ω	Steps	Time	Steps	Time	Steps	Time	Time	Time
gen200_p0.9_44	0.90	44	0	0.033	10789	0.134	156	**0.004**	8.3	34
gen200_p0.9_55	0.90	55	61	0.037	28616	0.294	49	**0.004**	9.2	74
gen400_p0.9_55	0.90	55	320	1.05	–	>3 h	1	**< 0.001**	1052	∞
gen400_p0.9_65	0.90	65	75	0.285	–	>3 h	2	**< 0.001**	285	∞
gen400_p0.9_75	0.90	75	160	0.238	–	>3 h	1579	**0.044**	5.4	∞
hamming10-2	0.99	512	0	25.0	1	0.015	1	**< 0.001**	24992	15
hamming8-2	0.97	128	0	0.013	1	0.001	1	**< 0.001**	13	1.0
hamming8-4	0.64	16	264	0.056	3025	0.022	2898	**0.012**	4.6	1.8
johnson16-2-4	0.77	8	3150	0.080	118237	0.081	114070	**0.068**	1.2	1.2
johnson8-2-4	0.55	4	0	< 0.001	13	< 0.001	12	**< 0.001**	1.0	1.0
johnson8-4-4	0.77	14	0	0.004	43	< 0.001	1	**< 0.001**	3.6	1.0
keller4	0.65	11	469	0.043	1477	**0.006**	1357	0.010	4.3	0.6
keller5	0.75	27	249409	127	–	>3 h	128634297	1364	0.1	∞
p_hat1000-1	0.25	10	1700	0.503	20306	**0.165**	19311	0.166	3.0	1.0
p_hat1000-2	0.49	46	51658	43.2	1987798	66.9	905774	**29.6**	1.5	2.3
p_hat1500-1	0.25	12	56404	4.64	95933	1.47	88109	**1.16**	4.0	1.3
p_hat1500-2	0.51	65	1517961	**1982**	108728522	6448	36361695	2237	0.9	2.9
p_hat300-1	0.24	8	15	0.011	247	**0.001**	177	0.002	5.7	0.5
p_hat300-2	0.49	25	21	0.032	413	0.012	212	**0.008**	4.0	1.5
p_hat300-3	0.74	36	506	0.313	55832	0.800	12175	**0.242**	1.3	3.3
p_hat500-1	0.25	9	366	0.066	694	0.015	617	**0.010**	6.6	1.5
p_hat500-2	0.51	36	254	0.165	8068	0.159	2257	**0.058**	2.8	2.7
p_hat500-3	0.75	50	28338	19.6	1420639	40.8	646861	**18.2**	1.1	2.2
p_hat700-1	0.25	11	559	0.121	2162	0.029	675	**0.026**	4.6	1.1
p_hat700-2	0.50	44	613	0.826	55703	1.57	18200	**0.534**	1.5	2.9
p_hat700-3	0.75	62	247906	**241**	15527523	725	5592597	262	0.9	2.8
san1000	0.50	15	646	0.421	15864	0.793	1	**< 0.001**	421	793

(*continued on next page*)

Table 1
(*continued*)

			IncMaxCLQ (Li *et al*., 2013)		BBMCX (San Segundo *et al*., 2015)		NEW		CLQ/NEW	BBMCX/NEW
	p	ω	Steps	Time	Steps	Time	Steps	Time	Time	Time
san200_0.7_1	0.70	30	6	0.016	357	0.003	1	**< 0.001**	16	3.0
san200_0.7_2	0.70	18	0	0.008	212	0.004	1	**< 0.001**	7.8	4.0
san200_0.9_1	0.90	70	0	0.032	436	0.015	1	**< 0.001**	32	15
san200_0.9_2	0.90	60	0	0.026	776	0.018	1	**< 0.001**	26	18
san200_0.9_3	0.90	44	93	0.044	1539	0.016	1	**< 0.001**	44	16
san400_0.5_1	0.50	13	55	0.032	482	0.011	1	**< 0.001**	32	11
san400_0.7_1	0.70	40	286	0.120	6890	0.155	1	**< 0.001**	120	155
san400_0.7_2	0.70	30	736	0.357	3624	0.092	1	**< 0.001**	357	92
san400_0.7_3	0.70	22	7364	2.12	35080	0.519	1	**< 0.001**	2121	519
san400_0.9_1	0.90	100	99	0.223	1259	0.041	2	**< 0.001**	223	41
sanr200_0.7	0.70	18	1366	0.193	15609	0.094	11792	**0.069**	2.8	1.4
sanr200_0.9	0.90	42	8029	**2.28**	759754	9.44	376883	4.62	0.5	2.0
sanr400_0.5	0.50	13	5041	0.491	26345	0.202	15826	**0.138**	3.6	1.5
sanr400_0.7	0.70	21	477635	90.2	5884650	53.9	5777785	**45.6**	2.0	1.2

Table 2
Performance of the algorithms considered over a data set of uniform random graphs. In bold, best times for each row. A *step* is a recursive call to the algorithm. The time limit was fixed at 3 h and is measured in seconds with millisecond precision. Times reported are averaged over 10 runs. The column with header p shows average density, header ω refers to the clique number and wang to the average clique number of the 10 runs.

				IncMaxCLQ (Li *et al.*, 2013)		BBMCX (San Segundo *et al.*, 2015)		NEW		CLQ/NEW	BBMCX/NEW
n	p	ω	ω_{avg}	Steps	Time	Steps	Time	Steps	Time	Time	Time
150	0.7	16–17	16.5	152	0.028	3176	0.018	2019	**0.015**	1.8	1.2
150	0.8	23	23	607	0.091	10570	0.068	7055	**0.049**	1.9	1.4
150	0.9	35–38	36.1	519	**0.129**	33793	0.278	25001	0.242	0.5	1.1
150	0.95	51–58	54.2	30	**0.021**	7003	0.092	2796	0.051	0.4	1.8
150	0.98	75–84	78.5	0	0.010	250	0.007	27	**0.002**	5.1	3.5
200	0.7	17–18	18	1560	0.210	18197	0.118	15001	**0.098**	2.1	1.2
200	0.8	25–26	25.2	6253	1.110	174890	1.141	121530	**0.839**	1.3	1.4
200	0.9	40–42	41.7	16201	**4.486**	1736303	18.21	929779	10.933	0.4	1.7
200	0.95	58–66	61.5	5008	**2.021**	1819358	30.927	1189164	23.269	0.1	1.3
200	0.98	90–103	93.2	2	**0.022**	2480	0.078	901	0.046	0.5	1.7
300	0.6	15–16	14.9	5090	0.661	42485	0.243	38571	**0.217**	3.0	1.1
300	0.7	20–21	20	34511	5.094	458336	3.124	354376	**2.499**	2.0	1.3
300	0.8	28–29	28.5	500066	101.964	12880189	113.196	8929109	**81.787**	1.2	1.4
500	0.4	10–11	10.4	3585	0.313	13676	0.099	12762	**0.099**	3.2	1.0
500	0.5	13	13.3	14800	1.974	90344	0.705	77226	**0.617**	3.2	1.1
500	0.6	17	17	172621	23.261	1293409	11.229	1001177	9.148	2.5	1.2
500	0.7	22–23	22.4	3514023	685.268	49894001	493.651	39122751	**390.419**	1.8	1.3
1000	0.2	7–8	7.3	752	0.195	1889	**0.042**	1446	0.043	4.5	1.0
1000	0.3	9–10	9.6	11844	1.006	33971	0.343	32586	**0.304**	3.3	1.1
1000	0.4	12	12	58408	9.232	309123	3.997	256411	**3.612**	2.6	1.1
1000	0.5	15	15	1729301	218.145	6492609	91.01	5921869	**83.784**	2.6	1.1
3000	0.1	6-7	6.4	98463	16.072	2551	0.231	2493	**0.225**	71.4	1.0
3000	0.2	9	9	823163	36.526	208690	4.707	199516	**4.067**	9.0	1.2
5000	0.1	7	7	438717	72.368	7340	1.44	4486	**1.383**	52.3	1.0
5000	0.2	9–10	9.1	16912464	613.796	1184296	82.851	1220961	**80.559**	7.6	1.0
10000	0.1	7–8	7.3	15117	46.069	409612	26.741	328864	**25.351**	1.8	1.1
15000	0.1	8	8	289917	186.520	1674351	158.769	1596434	**152.025**	1.2	1.0

idea in which only a partial colouring is computed for each subproblem. Consequently, BBMCX$_S$ computes bounds faster than BBMCX, but its pruning can be less effective. We then describe a second algorithm BBMCXR$_S$ which efficiently enhances BBMCXs with *recolouring*. The latter is a linear procedure which attempts to improve the sequential approximate-colour bound. Reported results show that BBMCXR$_S$ captures structure better, on average, than previous infra-chromatic solvers.

We would also like to highlight the template design of the algorithms presented in this work. We believe this will make it easier for other researchers to develop new variants, either by switching between frameworks, or by adding a new pruning scheme in a hierarchical fashion. Ongoing work is the enhancement of the standard, full colour, bound with a combined *recolouring* + *infra-chromatic* filter.

Acknowledgements This work is funded by the Spanish Ministry of Economy and Competitiveness (grant NAVEGASE: DPI 2014-53525-C3-1-R). Mikhail Batsyn, Alexey Nikolaev, and Panos M. Pardalos are supported by the Laboratory of Algorithms and Technologies for Network Analysis, National Research University Higher School of Economics. We would also like to thank Jorge Artieda for his help with the experiments and Chu-Min Li for providing the source code of IncMaxCLQ.

References

Balas, E., Yu, C.S. (1986). Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15(4), 1054–1068.

BBMC releases. https://www.biicode.com/pablodev/examples. Accessed 1 February 2016.

BITSCAN C++ library. https://www.biicode.com/pablodev/bitscan. Accessed 1 February 2016.

Butenko, S., Chaovalitwongse, W., Pardalos, P. (eds.) (2009). *Clustering Challenges in Biological Networks*. World Scientific, Singapore.

Carraghan, R., Pardalos, P.M. (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9, 375–382.

Eblen, J., Phillips, C., Rogers, G., Langston, M. (2012). The maximum clique enumeration problem: algorithms, applications, and implementations. *BMC Bioinformatics*, 13, S5.

GRAPH C++ library. https://www.biicode.com/pablodev/graph. Accessed 1 February 2016.

Johnson, D., Trick, M. (Eds.) (1996). *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 26, American Mathematical Society.

Karp, R.M. (1972). *Reducibility among Combinatorial Problems*. Plenum Press, New York, pp. 85–103.

Konc, J., Janezic, D. (2007). An improved branch and bound algorithm for the maximum clique problem. *MATCH Communications in Mathematical and in Computer Chemistry*, 58, 569–590.

Konc, J., Janezic, D. (2010). Algorithm for detection of structurally similar protein binding sites by local structural alignment. *Bioinformatics*, 26, 1160–1168.

Leighton, F.T. (1979). A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, 84(6), 489–506.

Li, C.M., Quan, Z. (2010). Combining graph structure exploitation and propositional reasoning for the maximum clique problem. In: *Proceedings of the XXII International Conference on Tools with Artificial Intelligence*, pp. 344–351.

Li, C.M., Zhiwen, F., Ke, X. (2013). Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem. In: *Proceedings of the XXV International Conference on Tools with Artificial Intelligence*, pp. 939–946.

Maslov, E., Batsyn, M., Pardalos, P. (2013). Speeding up branch and bound algorithms for solving the maximum clique problem. *Journal of Global Optimization*, 59, 1–21.

Matula, D.W., Beck, L.L. (1983). Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the Association for Computing Machinery*, 30(3), 417–427.

Prosser, P. (2012). Exact algorithms for maximum clique: a computational study. *Algorithms*, 5(4), 545–587.

San Segundo, P., Artieda, J. (2015). A novel clique formulation for the visual feature matching problem. *Applied Intelligence*, 43(2), 325–342.

San Segundo, P., Rodriguez-Losada, D. (2013). Robust global feature based data Association with a sparse bit optimized maximum clique algorithm. *IEEE Transactions on Robotics*, 29, 1332–1339.

San Segundo, P., Tapia, C. (2010). A new implicit branching strategy for exact maximum clique. In: *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, Vol. 1. IEEE, pp. 352–357.

San Segundo, P., Tapia, C. (2014). Relaxed approximate coloring in exact maximum clique search. *Computers & Operations Research*, 44, 185–192.

San Segundo, P., Rodriguez-Losada, D., Matia, F., Galan, R. (2010). Fast exact feature based data correspondence search with an efficient bit-parallel MCP solver. *Applied Intelligence*, 32(3), 311–329.

San Segundo, P., Rodriguez-Losada, D., Jimenez, A. (2011a). An exact bit-parallel algorithm for the maximum clique problem. *Computers & Operations Research*, 38(2), 571–581.

San Segundo, P. Matia, F., Rodriguez-Losada, D., Hernando, M. (2011b). An improved bit parallel exact maximum clique algorithm. *Optimization Letters*, 7(3), 467–479.

San Segundo, P., Nikolaev, A., Batsyn, M. (2015). Infra-chromatic bound for exact maximum clique search. *Computers & Operations Research*, 64, 293–303.

Tomita, E., Seki, T. (2003). An efficient branch and bound algorithm for finding a maximum clique. In: Calude, C., Dinneen, M., Vajnovszki, V. (Eds.), *Discrete Mathematics and Theoretical Computer Science*, *LNCS*, Vol. 2731, pp. 278–289.

Tomita, E., Sutani, Y., Higashi, T., Takahashi, S., Wakatsuki, M. (2010). A simple and faster branch-and-bound algorithm for finding a maximum clique. In: *Lecture Notes in Computer Science*, Vol. 5942, 191–2030.

Žilinskas, J. (2011). Copositive Programming by Simplicial Partition. *Informatica*, 22(4), 601–614.

Žilinskas, J., Dür, M. (2011). Depth-first simplicial partition for copositivity detection, with an application to MaxClique. *Optimization Methods and Software*, 26(3), 499–510.

P. San Segundo serves as Associate Professor at the Polytechnic University of Madrid (UPM) and fellow researcher at the Centre of Automation and Robotics (CAR). His main research interests lie in Artificial Intelligence (search problems) and combinatorial optimization. He is the developer of recent leading exact algorithms PASS for the vertex coloring problem and BBMC for the maximum clique problem. Moreover he is also an International Chess GrandMaster and former member of the Spanish Olympic Team for many years.

A. Nikolaev is a research assistant at Laboratory of Algorithms and Technologies for Network Analysis, National Research University Higher School of Economics, Russia. His research interests include discrete optimization and machine learning.

M. Batsyn is a leading research fellow in the Laboratory of Algorithms and Technologies for Network Analysis (LATNA), National Research University Higher School of Economics, Russia. His research interests include discrete optimization problems, branch-and-bound algorithms, classical heuristics and meta-heuristic approaches, real-life optimization problems.

P.M. Pardalos is a renowned world leading expert in global and combinatorial optimization. His recent research interests include network design problems, optimization in telecommunications, e-commerce, data mining, biomedical applications, and massive computing. He serves as Distinguished Professor of Industrial and Systems Engineering at the University of Florida. Moreover, he is also the Paul and Heidi Brown Preeminent Professor in Industrial & Systems Engineering.

Pagerintas infrachromatinis rėžis tiksliai didžiausios klikos paieškai

Pablo SAN SEGUNDO, Alexey NIKOLAEV, Mikhail BATSYN, Panos M.PARDALOS

Šiuo straipsniu pagerinamas infrachromatinis rėžis, naudojamas tiksliam šakų ir rėžių didžiausios klikos sprendikliui BBMCX kaip viršutinis klikos dydžio rėžis kiekvienam daliniam uždaviniui. Infra-chromatinis rėžis ieško spalvų poaibių trijulių, negalinčių turėti 3-klikų. Toks rėžis yra griežtesnis negu gaunamas dažnai naudojamų apytikslio spalvojimo algoritmų, nes gali būti mažesnis už chromatinį skaičių. Pristatyti rezultatai rodo, kad mūsų algoritmas su naujuoju rėžiu gerokai lenkia pranašius algoritmus struktūriniams ir atsitiktiniams grafams.