Appl Intell (2016) 45:868–880
DOI 10.1007/s10489-016-0796-9

# Improved initial vertex ordering for exact maximum clique search

**Pablo San Segundo[1] · Alvaro Lopez[1] · Mikhail Batsyn[2] · Alexey Nikolaev[2] · Panos M. Pardalos[2,3]**

Published online: 24 May 2016


**Abstract** This paper describes a new initial vertex ordering procedure NEW_SORT designed to enhance approximate-colour exact algorithms for the maximum clique problem (MCP). NEW_SORT considers two different vertex orderings: degree and colour-based. The degree-based vertex ordering describes an improvement over a well-known vertex ordering used by exact solvers. Moreover, colour-based vertex orderings for the MCP have been traditionally considered suboptimal with respect to degree-based ones. NEW_SORT chooses the "best" of the two orderings according to a new evaluation function. The reported experiments on graphs taken from public datasets show that a leading exact solver using NEW_SORT —and further enhanced with a strong initial solution— can improve its performance very significantly (sometimes even exponentially).

**Keywords** Maximum clique · Branch-and-bound · Approximate colouring · Combinatorial optimization

✉ Pablo San Segundo
pablo.sansegundo@upm.es

1 Centre for Automation and Robotics (UPM-CSIC), C/ Jose Gutiérrez Abascal, 2; 28006, Madrid, Spain

2 Laboratory of Algorithms and Technologies for Networks Analysis, National Research University Higher School of Economics, 136 Rodionova, Niznhy Novgorod, Russia

3 Center for Applied Optimization, University of Florida, 303 Weil Hall, Gainesville, FL 32611, USA

## 1 Introduction

A simple, undirected graph $G = (V, E)$ is defined by a set of vertices $V = \{v_1, v_2, \cdots, v_n\}$ and a set of edges $E$ made up of pairs of distinct vertices ($E \subseteq V \times V$). A *clique* in graph $G$ is a complete subgraph, that is, a subgraph in which vertices are pairwise adjacent. In this work, we consider the *maximum clique problem* (MCP), which asks for a clique of the largest cardinality in the graph. The size of a maximum clique is called the *clique number* of the graph and is usually denoted as $\omega(G)$.

The MCP is a well known and deeply studied NP-hard problem in graph theory. Moreover, it has found applications in many different fields, such as data association problems in bioinformatics and computational biology [1–3], computer vision [4], and robotics [5]. Such association problems may be reduced to the MCP in a *correspondence graph*, which subsumes the matching criteria between the two entities involved. With the upsurge of Web technologies, cliques have also been applied to capture the structure of massive networks. For example, in social networks a clique can identify a group of cooperating agents (e.g. a terrorist cell); in the World Wide Web, cliques or quasi-cliques can help detect frequently visited pages concerning a certain topic. Clique kernels can also help to identify clusters.

Relevant definitions and notation used in the paper are the following:

– $G[W] = (W, E[W])$ : a subgraph of graph $G$ induced by a vertex set $W \subseteq V$.
– $N(u) = \{v \in V | (u, v) \in E\}$ : the *neighbour set* of vertex $u$ in graph $G$, that is, the set of vertices adjacent to $u$. Notation may include a vertex set as a subscript (e.g.

$N_W(u)$) to refer to the neighbourhood in the induced subgraph $G[W]$.
– $\deg(u) = |N(u)|$: the *degree* of vertex $u$.
– *k-colouring*: an assignment of $k$ different numbers (colours) to every vertex of graph $G$ such that adjacent vertices have different colours, that is, $u \in N(v) \Rightarrow c(u) \neq c(v)$. A k-colouring partitions the vertex set $V$ into $k$ disjoint colour sets $C_1, C_2, \ldots, C_k$, also called colour *classes*. Each colour set $C_1, C_2, \ldots, C_k$, is an independent set, that is, a set of pairwise non-adjacent vertices.
– $\chi(G)$: the *chromatic number* of graph $G$, that is, the minimum number $k$ of colours required to colour $G$.
– *greedy sequential colouring* (SEQ): a colouring heuristic which sequentially assigns the lowest possible colour number to each vertex.
– $w(v_i) = |N(v_i) \cap \{v_1, \ldots, v_{i-1}\}|$: denotes the *width* at the $i$-th vertex in a sequence, that is, the number of vertices in $N(v_i)$ that precede $v_i$. The *width of an ordering* is the maximum width at any of its vertices.
– *degeneracy* (or *width)* of $G$: the minimum width of any ordering of $V$.
– *minimum-degree-last* ordering of vertices: a vertex ordering of minimum width obtained by iteratively *removing* vertices with minimum degree and *placing them in reverse order* in the new ordering.
– $\sigma(v) = \sum_{u \in N(v)} \deg(u)$: the neighbourhood *support* of $v$, that is, the sum of its neighbours' degrees.

### 1.1 Exact branch and bound algorithms

In the literature, there are many different approaches to solving the MCP exactly. Most successful exact solvers belong to the family of branch-and-bound algorithms that employ approximate-colour bounds [6–19]. Fahle's algorithm [7] is possibly the first solver of this type.

Exhaustive enumeration can be traced back to the classic Bron-Kerbosch algorithm [19]. Exact solvers keep track of a growing clique in $S$ and a candidate set of vertices $U$ that can enlarge $S$. At each step, a single vertex $v$ is selected from $U$ to build a bigger clique in $S$ and create a new, smaller subproblem, with a set of candidates $N_U(v)$. Leaf nodes of the search tree correspond to maximal cliques and during enumeration, they are checked to see whether their size is greater than the incumbent solution stored in a global variable $S_{max}$. Every time a bigger clique is found, it is written to $S_{max}$.

The basic branch-and-bound approach for the MCP can be traced to Carraghan and Pardalos in [22]. Approximate colour bounds for a maximum clique achieve a good compromise between tightness and computational effort. Proposition 1 provides theoretical justification for this, and may be derived trivially from [20].

**Proposition 1** *Any k-colouring of a graph G gives an upper bound on its clique number ($\omega(G) \leq \chi(G) \leq k$).*

In most of the effective exact maximum clique approximate-colour algorithms for the MCP, the greedy sequential colouring heuristic SEQ is employed to colour each subproblem. SEQ is a constructive heuristic that iteratively assigns the smallest possible colour to every vertex such that no conflicts with the already coloured vertices occur. It has a worst-case running time of $O(n^2)$.

Relevant recent improvements reported in the literature for exact maximum clique algorithms that employ approximate-colour bounds are (in chronological order):

– *Branching on maximum colour*: at each step (a recursive call of the algorithm), vertices are selected for branching in non-increasing order of their colour numbers. This was first described in algorithm MCQ [8].
– *Recolouring*: an additional computation which aims at reducing the size of the colouring obtained by SEQ, but increases its complexity linearly to $O(n^3)$. It was first described in algorithm MCS [9].
– *Static ordering of vertices*: vertices in every subproblem are always sorted in the order determined at the beginning of the search. This was first described independently both in MCS and in the bit-parallel kernel of the BBMC family of algorithms [10, 11].
– *Bitstring encoding of the MCP* [10–12, 15]: the BBMC family of algorithms represents vertex sets, as well as the input graph, via bitstrings. The advantage is that critical operations related to child problem generation and bound computation are performed more efficiently using bitmasks.
– *Selective colouring:* a partial SEQ colouring in which only the subset of vertices to be pruned in the child subproblem is coloured. It was first described in BBMCL [12] (the 'L' stands for seLective).
– *Strong heuristic for a 'good' initial solution*, as described in [17].
– *Infra-chromatic bound*: a bound tighter than the one obtained by SEQ. This bound can possibly be lower than the chromatic number of the input graph. In MaxCLQ [13, 14], the authors proposed one such bound based on reducing the maximum clique problem determined by each coloured subgraph to the partial maximum satisfiability problem. The term *infra-chromatic* first appeared in [15], where the BBMCX algorithm is described. BBMCX shares the bitstring BBMC kernel and implements an infra-chromatic bound by looking for triplets of colour sets in which there are no vertices

that can form a triangle. For each such triplet, denoted *inconsistent*, the bound is decremented from 3 to 2. BBMCX is currently the fastest published algorithm of the BBMC family for dense graphs.

Table 1 summarizes the majority of algorithms described in this section, together with their most relevant properties.

This work describes two initial orderings of vertices that are efficient for successful approximate-colour solvers, such as MCS or BBMCX, with the exception of MaxSAT-based MaxCLQ.

Related to branching on maximum colour and static ordering is the fact that *the initial sorting of vertices is well known to have a significant impact on the size of the MCP search tree*. We discuss this issue in the next subsection, as it is very much concerned with the contribution of this work.

### 1.2 Initial ordering of vertices

A well-known initial sorting strategy for exact MCP solvers is to branch on vertices with the smallest degree at the root node. The idea is similar to branching on variables with a small number of values used in constraint satisfaction problems. In practice, *vertices with the smallest degree are placed last* in $V$ and branching in all subproblems, including the root node, is done by selecting vertices in reverse order.

The most successful initial sorting strategies for exact MCP algorithms reported in the literature are the following:

- *Minimum width* (MW): a *minimum-degree-last* ordering (ties broken randomly or in natural order), which can be traced back to [22] in connection with the MCP. As explained previously, it is a degenerate ordering achieved by removing, at each iteration, the vertex with the minimum degree and placing it in reverse order in the new ordering.
- *Minimum width with tie break by minimum support* (MWS): Similar to MW but with a tie-break strategy: it selects the vertex with the minimum support from the set of vertices with the same degree.

A lighter variant for computing MWS was brought to our attention in a personal communication [23] and will be referred to as MWSS (Minimum Width with Static Support). It is similar to MWS, but instead of recalculating neighbour support at each step it uses the vertex support

**Table 1** A number of relevant exact maximum clique solvers in chronological order

| Name | Search strategy |
|---|---|
| *Bron-Kerbosch* [19] | Lists all maximal cliques. The worst-case complexity is $O(3^{n/3})$ for this problem. |
| *EA* [22] | Describes the basic branch-and-bound strategy (EA stands for Enumerative Algorithm). |
| *Cliquer* [6] | Examines subproblems incrementally, in reverse order with respect to *EA*. At present, it is still a leading algorithm for the weighted maximum clique problem. |
| $\chi + DF$ [7] | The first time approximate colouring has been described as bound. |
| *MCQ* [8] | The first time branching on maximum colour has been described. |
| *MCS* [9] | Uses a fixed order of vertices in all subproblems. Describes *recolouring* for the first time. |
| *BBMC* [10] | The first efficient bit-parallel exact solver. Uses a fixed order of vertices in all subproblems (found independently of MCS). |
| *BBMCL* [12] | Colours only a subset of vertices in each subproblem. |
| *MaxCLQ* [13, 14] | Reduces each subproblem to a partial maximum satisfiability problem. Uses logical inferences to improve a colour bound. |
| *BBMCX* [15] | Looks for triplets of colour sets that are triangle-free to improve a colour bound. The first time the term *infra-chromatic* has been used to refer to a bounding strategy. |

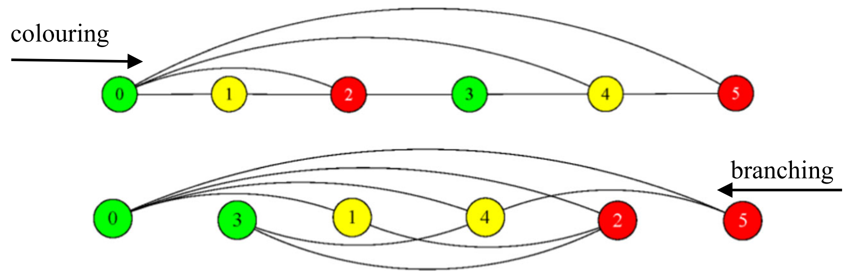



**Fig. 1** Colouring and vertex selection directions in each subproblem

determined by the initial ordering in every iteration. MWSS is very useful in graphs of high order, in which the computational cost of MWS is high. By default, support tiebreak in this paper always refers to MWS. MWSS will be explicitly mentioned when disambiguation is required.

Initial sorting by degree has been the standard choice of successful approximate-colour exact algorithms for the MCP. Recently, a colour-based ordering was described in [18] as a possible enhancement of the MaxCLQ algorithm. However, the specific impact of the ordering was not analysed.

In Section 2 of this paper, we describe a new initial sorting procedure, DEG_SORT, which improves standard MW/MWS degree-based orderings. Section 3 starts by describing a colour-based initial sorting procedure COLOUR_SORT, based on [18], and explains why it can also be successful for MCS or the BBMC family of algorithms. The final part of the section describes the NEW_SORT algorithm proposed in this work. NEW_SORT selects DEG_SORT or COLOUR_SORT according to a new evaluation function. Section 4 covers the experiments and validation. Finally, Section 5 presents the conclusions and future work.

## 2 Improved degree-based initial sorting

Branching on vertices with the highest colour was first proposed in MCQ [8] and, since then, has been applied by most successful MCP exact solvers. In practice, as mentioned previously, vertices are sorted in a *highest-colour-last* fashion and taken *in reverse order* in every subproblem. Figure 1 depicts the control flow: vertices are coloured by greedy SEQ according to the initial ordering (Fig. 1, top), sorted according to non-decreasing colour number, and then selected in reverse order (Fig. 1, bottom).

### 2.1 Analysis of largest-first vertex colouring heuristic

To evaluate the quality of the bounds obtained by direct implementation of the flow in Fig. 1 (very much related to the proposed new sorting heuristic), we consider the *Largest-First* (LF) decision heuristic for greedy vertex colouring of Welsh and Powell. In [24], they proved that given a non-increasing degree ordering of vertices (i.e. $\deg(v_1) \geq \deg(v_2) \geq \ldots \geq \deg(v_n)$), SEQ would always produce not more than $\max_{i \in V} \min\{i, 1+\deg(v_i)\}$ colours. This is known as the *Welsh and Powell bound*. We will refer to the

**Table 2** Comparison between *Largest-First* and *Smallest-First* colouring heuristics for a number of structured (see Appendix) and uniform random graphs

| n | SF | LF [24] | %imp | Family | SF | LF [24] | %imp |
|---|---|---|---|---|---|---|---|
| 100 | 25.0 | 21.9 | 12.4 | *C* | 81.0 | 75.5 | 6.8 |
| 150 | 33.7 | 30.0 | 11.0 | *MANN* | 175.0 | 180.0 | ?2.9 |
| 200 | 41.9 | 37.6 | 10.4 | *brock* | 75.8 | 71.0 | 6.3 |
| 250 | 49.7 | 44.9 | 9.7 | *c-fat* | 52.9 | 52.1 | 1.4 |
| 300 | 57.1 | 52.0 | 8.8 | *dsjc* | 65.5 | 60.3 | 8.0 |
| 350 | 64.5 | 58.9 | 8.7 | *frb30-15* | 67.8 | 49.4 | 27.1 |
| 400 | 71.5 | 65.6 | 8.3 | *gen200* | 81.0 | 70.0 | 13.6 |
| 450 | 78.5 | 72.2 | 8.0 | *p-hat* | 106.8 | 74.4 | 30.3 |
| 500 | 85.5 | 78.7 | 7.9 | *san* | 61.1 | 58.3 | 4.6 |
| 1000 | 149.4 | 140.6 | 5.9 | *sanr* | 76.5 | 68.8 | 10.1 |

Each cell reports average colour sizes for each case. In the case of random graphs $(n, p)$ we consider, for each value of $n$, densities from 0.1 to 0.9. Colour sizes for each density are averaged over 50 runs. In the case of structured instances, average colour sizes are reported for typical members of each family. The "%imp" column shows the improvement of LF over SF as a percentage.

opposite (*bad*) ordering $\deg(v_1) \leq \deg(v_2) \leq \ldots \leq \deg(v_n)$ as *Smallest-First* (SF).

The key idea of LF is to assign colour numbers to *the most conflicting* vertices early in the hope that those remaining will require a small number of colours (ideally, not different from those used in the early stages). Moreover, Observation 1 is widely accepted (see for example [25]) and many recent exact MCP solvers apply some variant of LF ordering for SEQ colouring [8–12, 15–17].

**Observation 1** *Greedy sequential colouring of vertices sorted according to the LF rule almost always produces tighter colourings than the Welsh and Powell bound.*

A qualitative measure of the impact of LF ordering in SEQ may be found in Table 2. There, LF is compared with its counterpart SF in structured and non-structured uniform random graphs. Note that we do not compare the number of colours in LF colouring with the chromatic number since it is impractical to compute the chromatic number in most of the graphs.

In the case of structured instances, the table reports average colour sizes for typical members of each family (*brock200_1* – *brock400_4*, *dsjc 500.1/5* – *dsjc 1000.1/5*, *MANN_a9* – *MANN_a27* etc.; see the Appendix for the full list). In the case of Erdös-Rényi random graphs $G(n, p)$, the table reports average colour sizes for different values of $n$ and density $p$ (we consider 50 instances for each graph type). Table 2 gives evidence that the LF rule produces tighter sequential colourings, on average, than the SF one: up to 12 % improvement for non-structured graphs and 30 % for structured graphs. The exception is the *MANN* family, in which SF actually improves the colouring. This may be explained by the high density ($p > 0.92$) of these particular graphs. We note that even a small bound improvement can produce an exponential reduction in the size of the maximum clique search tree.

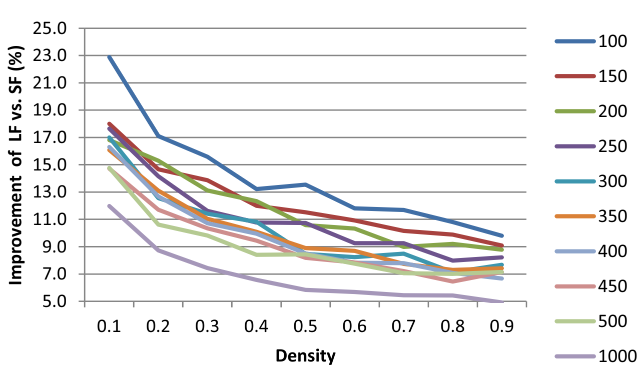



**Fig. 2** Impact of *Largest-First* sequential greedy colouring on Erdös-Réényi graphs of different sizes and densities. The Y-axis refers to the improvement with respect to Smallest-First as a percentage. Each line corresponds to a different graph order

Another interesting result to be derived from Table 2 is Observation 2, where the term *benefit* refers to the gap between LF and SF as a percentage.

**Observation 2** *The benefit of LF ordering for SEQ colouring diminishes with the growth of graph size and density in the case of uniform random graphs.*

Figure 2 corresponds with the data in Table 2 but includes density information for each graph order considered. Observation 2 is captured by the fact that *lines in the line chart are aligned by increasing graph size from top to bottom in the figure*. Lines cross for some neighbour sizes and different densities [e.g. (200, 0.1) shows a 16.81 % improvement whereas (250, 0.1) shows a superior 17.65 % improvement], but the trend is clearly there. We are not aware of this fact being reported elsewhere and consider Observation 2 as an additional contribution of the paper.

We propose the following intuition as an explanation. Let us consider the cases in which any sequential ordering mistakenly uses colour $\chi(G)+1$ for some vertex $v$, where $\chi(G)$ is the chromatic number of the coloured graph $G$. Such a case is shown in Fig. 3, in which vertex $v$ has $\chi$ neighbours with colours $1, \ldots, \chi$ and has to be coloured with colour $\chi + 1$.

In the case of the LF sequential colouring, this happens when *vertex $v$ has a smaller degree than each of these $\chi$ neighbours*. For a better understanding, we present several such cases in Fig. 4. Colours are shown with numbers near vertices. In the first graph, vertex $v$ has two neighbours and colour 3, although the chromatic number is 2; in the second graph, it has three neighbours and colour 4; in the third, it has four neighbours and colour 5.

We now show informally that the *bad* case depicted in Fig. 3 is more likely to occur when the number of edges

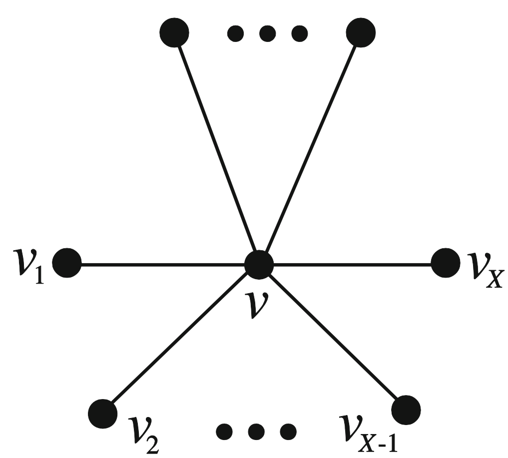



**Fig. 3** An example where vertex $v$ has $\chi$ neighbours $v_1, \ldots, v_\chi$ with colour numbers $1, \ldots, \chi$ respectively

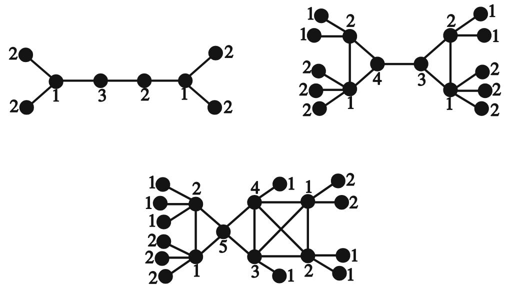


**Fig. 4** Examples in which Largest-First sequential colouring uses $\chi + 1$ colours

per vertex is small. This would explain the decrease in performance of LF with graph order as well as density. For a given graph $G = (V, E)$, let us consider an increase in the ratio $|E|/|V|$ and thus an increase in average and maximum degrees. This also results in an increment of the clique number $\omega$ and the chromatic number $\chi$ because $\chi \geq \omega$. In this scenario, the probability of the case shown in Fig. 3 decreases mainly for two reasons: first, because the degree of vertex $v$ cannot be less than $\chi$, and therefore its expected degree increases faster than the expected minimum degree of its $\chi$ neighbours; second, because the probability of these $\chi$ neighbours *all* having degree greater than $\deg(V)$ decreases as the number $\chi$ of these neighbours increases.

What we have presented is just an intuitive explanation of Observation 2. We believe that attempting to provide rigorous proof is, at this point, impractical. It would probably require a big theorem for a relatively simple result.

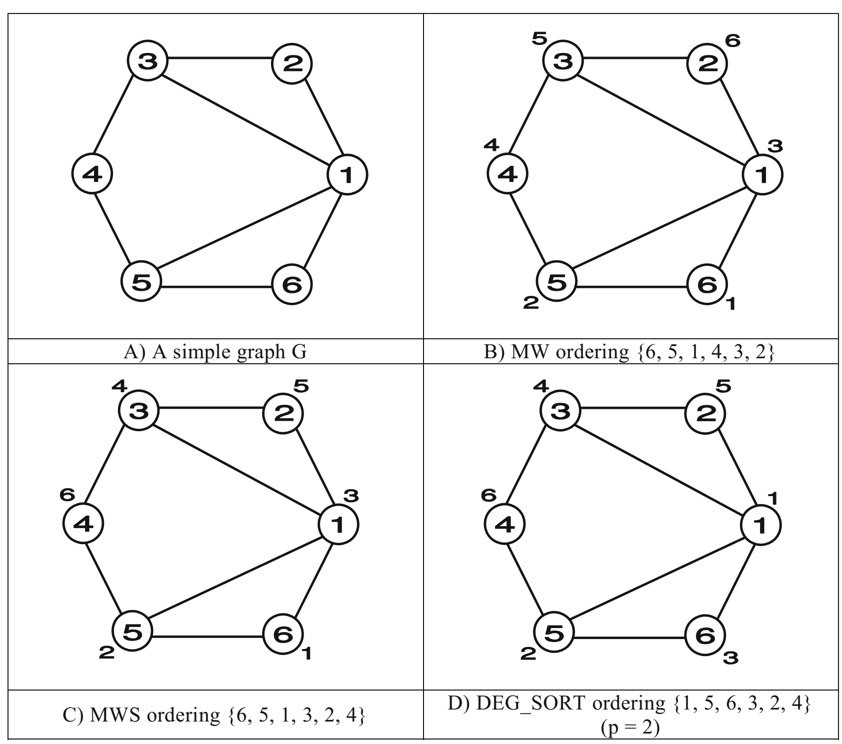
## 2.2 Sorting a fraction of vertices by non-increasing degree

Having established the relevance of LF sorting in sequential colouring, we now proceed to describe a new sorting procedure for exact maximum clique algorithms. In MCQ [8], the colour ordering required for branching (Fig. 1, bottom) is inherited in child subproblems. As a consequence, SEQ is given a suboptimal (non-LF) ordering and its pruning ability is diminished. A first alternative to improve this situation, and described in [16], was to reorder vertices by non-increasing degree prior to colouring (i.e. explicit LF), but its computational cost is high. The paper also described a way to *selectively* apply this strategy in the shallower levels of the search tree.

A better compromise (currently considered the best approach) is to use a *static ordering* in all subproblems. As mentioned in the introductory section, this decision heuristic was first proposed independently in [9] and [10] and is currently used by state-of-the-art BBMC and MCS solvers. In *static ordering*, vertices in every subproblem are always kept in the same relative order as determined initially. Specifically, the pruning ability of static ordering is high in the shallow levels of the search tree and degrades with depth, as subproblems become smaller and the initial sorting is gradually lost.

Related to the colour flow in Fig. 1, both initial vertex ordering strategies MW and MWS described in Section 1.2 are reasonably consistent with LF greedy colouring, in the sense that vertices with high degrees

**Fig. 5** Different initial vertex orderings for the MCP. The small numbers near the vertices in B, C, and D indicate their new positions



are *implicitly placed first in V* and colouring proceeds from first to last. However, vertices are *actually placed following a smallest-degree-last* strategy, which can differ considerably from an explicit *highest-degree-first* sorting because both MW and MWS are degenerate orderings.

It is easy to see this effect with the example depicted in Fig. 5. Figure 5A shows a simple graph $G$ in which vertices are numbered according to an initial default ordering that uniquely identifies them in the rest of the figures. This ordering will also determine tiebreaks when required. From the perspective of the control flow in Fig. 1, vertices are coloured in natural order (i.e. starting from vertex {1} and going anti-clockwise) and selected in reverse order (i.e. starting from {6} and going clockwise).

Figure 5B presents the minimum width ordering (MW) of the graph, and Fig. 5C the minimum width ordering with vertex support (MWS). The difference between them lies in the support of vertices {2} and {4}, which have both the same degree ($\deg(2) = \deg(4) = 2$). Ties are broken by vertex number for MW, so vertex {2} is picked first (and placed last) in the new ordering. In the case of MWS, $\sigma(2) = 7$, whereas $\sigma(4) = 6$, so vertex {4} is the one placed at the end. After removing {4}, two triangles appear: {1, 2, 3} and {1, 5, 6}; vertices {2, 3, 5, 6} all have minimum degree and support, so vertex {2} is selected in second place and so on.

Examining the resulting MW and MWS orderings from the perspective of the control flow in Fig. 1, it is clear that vertices are not sorted by non-increasing degree at the head of the ordering. In particular, the vertex with the highest degree {1} ($\deg(1) = 4$) comes in third place in both cases. The reason for this lies in the degenerate ordering, which iteratively *removes* each sorted vertex and thus reduces the degree of the remaining vertices to their core number. In the example, vertices {1, 5, 6} are the last remaining vertices for both MW and MWS (a three-clique). The latter graph is obviously also regular, so all vertices have the same degree and are sorted in reverse order of their numbers. As a consequence, vertex {1} is misplaced.

In the light of the above considerations, we propose an improved initial sorting procedure DEG_SORT, which can be seen as a repair mechanism for MW and MWS with respect to (maximum) degree at the head of the ordering. DEG_SORT takes as input MWS and sorts, according to non-increasing degree, a subset of the first $k$ vertices $v_1, v_2 \cdots, v_k$ (vertices with the same degree are taken according to their number). This second ordering is absolute (not degenerate) since it is directed to be as close as possible to LF in the subproblems that appear in the shallow levels of the search tree. The remaining $n - k$ vertices are not modified and remain sorted by minimum width with vertex support. Figure 5.D shows the ordering obtained by DEG_SORT in the example: vertex {1} with the highest degree is swapped with vertex {6} and placed first in the list.

Parameter $k$ (the number of vertices reordered by DEG_SORT) should be neither too small (and thus with low impact) nor too big (the original minimum width ordering would be lost). Rather than using $k$ as a tuning parameter, we consider a new parameter $p$ related to the total number of vertices and define it as follows:

$$p = \left\lfloor \frac{|V|}{k} \right\rfloor, \; p = \{2, 3, \ldots\}$$

In practice, DEG_SORT performs best when $p$ ranges between 2 (50 % of the vertices) and 10 (10 % of the vertices). In non-structured Erdös-Rényi graphs, the best results on average appear when $p$ is set to 3. In the case of structured graphs, they are obtained when $p$ is set to 4, but tuning is recommended in both cases whenever possible.

## 3 Colour-based initial ordering of vertices

### 3.1 Preliminaries

As explained in previous sections, an initial ordering of vertices based on degree is well known to reduce the size of the search tree in exact maximum clique search. It is also employed by successful modern algorithms such as BBMC and MCS. The logic behind it is to minimize branching in the first level of the tree. Moreover, BBMC and MCS preserve the ordering in every other subproblem as well (to improve the bound obtained by SEQ (see Fig. 1), so the benefits of a good initial ordering also propagate down the search tree to a certain depth.

In [18], the possibility of sorting vertices initially according to a colouring of the graph $C(G) = C_1, C_2, \ldots, C_k$, was described. The intuition is that it should somehow prune the maximum clique search space effectively in graphs where $k$ is a good bound on the clique number, but this was not analysed systematically in the original paper. Interestingly, the current implementations of BBMCX and MCS spend little effort in computing upper bounds on maximum clique at the root node. A typical strategy is to assign to a vertex as colour number the minimum value between its index and maximum graph degree. The above considerations motivate a systematic study of colour-based initial sorting.

The next subsection describes the sorting procedure COLOUR_SORT, which is based on [18] with additional refinements. In Subsection 3.3, we give additional explanations as to why COLOUR_SORT can be successful for BBMCX or MCS with an example. Finally, the last subsection describes the new sorting algorithm NEW_SORT, which is the main contribution of this work.

### 3.2 The colour-based sorting algorithm

COLOUR_SORT is described in Algorithm 1. The main computation is a variant of the constructive recursive-largest-first (RLF) colouring heuristic, which was first described in [26]. RLF computes colour classes one at a time and does not proceed with another colour until no more vertices can enlarge the current one. In the original paper, the assignment is implemented in the following way: when a new colour class $C_k$ is opened, set $W_1$ contains all remaining uncoloured vertices and set $W_2$ is empty. Iteratively, a vertex $v \in W_1$ is selected, added to $C_k$, and removed from $W_1$. If $v$ has any neighbours, they are also removed from $W_1$ and placed in $W_2$. The assignment of vertices proceeds until $W_1 = \phi$. The selection of vertices is based on degree. The first vertex is the one with maximum degree in $G[W_1]$ and the rest of vertices are those with maximum degree in $G[W_2]$. Once $W_1$ becomes empty, the next colour class is built.

COLOUR_SORT orders vertices in $V$ according to the colour classes obtained by RLF. The specific variant used takes into account two factors:

- A strong exact maximum clique algorithm is available, in this case BBMCX.
- The graph to be ordered is expected to be dense, since finding its clique number presents a challenge.

The actual RLF variant used by COLOUR_SORT computes each new colour set as an independent set (a maximum clique in the complement graph $\bar{G}$) (steps 2 to 7). Once a colour set is produced, its vertices are placed in order in $O_{color}$ and removed from $\bar{G}$. COLOUR_SORT then proceeds with a new colour set until no more vertices are left in $\bar{G}$.

**Algorithm 1** An initial colour-based ordering of vertices for efficient maximum clique search

*Input*: A simple graph $G = (V, E)$
*Output*: A vertex ordering $O_{color}$ and a number $k$ related to its computation
*Initial values*: $O_{color} \leftarrow \phi, k \leftarrow 0, W \leftarrow V$

COLOUR_SORT($G$)
1. $\bar{G} \leftarrow$ compute the complement graph of $G$
2. **repeat until** $W = \phi$
3. $\quad U \leftarrow$ a maximum clique of $\overline{G[W]}$
4. $\quad O_{color} \leftarrow O_{color} \cup U$
5. $\quad W \leftarrow W \backslash U$
6. $\quad k \leftarrow K + 1$
7. **endrepeat**
8. **return** $(O_{color}, k)$

### 3.3 An example

To see why COLOUR_SORT can be beneficial for successful approximate-colour algorithms, we will use the coloured graph $G$ depicted in Fig. 6. We assume $G$ to be a subproblem, close to a leaf node, of a maximum clique search tree. The output of SEQ for the graph is $C_1 = \{1, 2\}$(green), $C_2 = \{3, 4\}$(yellow), and $C_5 = \{5\}$(cyan), as shown. The figure also indicates the colour threshold $k_{\min}$ (the difference between the size of the best clique found so far $|S_{\max}|$ and the size of the clique being built in the branch $|S|$) for the subproblem, which is 3. This implies that all vertices belonging to colour classes below this threshold (in the example, sets $C_1$ and $C_2$) will be pruned in any derived child node (for a more detailed description of the threshold, see [12] amongst others).

In algorithms such as BBMC or MCS, pruning the search space can be seen as a technique that accumulates as many vertices as possible *behind* the $k_{min}$ threshold. There are three main alternatives to achieve this:

I *Incrementing the colour threshold* $k_{min}$, or, alternatively, moving the dotted line to the right: this can be done by finding good solutions early, either by making good branching choices or by computing a strong initial solution. Note that the latter can produce very effective pruning, since it increases $|S_{\max}|$ in the shallow levels of the search tree.

II *Shifting vertices from the right to the left of the threshold*: this can be achieved with techniques such as recolouring or infra-chromatic pruning. In the example, BBMCX detects that the induced subgraph $G[C_1 \cup C_2 \cup C_3]$ is triangle-free and reduces the bound from 3 to 2, so that $\{5\}$now falls below the threshold.

III *Improving the quality of the greedy SEQ colouring*, that is, changing its output to produce colour classes $C_i$, $i < k_{\min}$, that are as large as possible.

The last point is especially relevant to explain why COLOUR_SORT could be successful for some graphs. SEQ is an oriented heuristic. If, in the example, the vertices were presented in the order $\{2\}$, $\{3\}$, $\{5\}$, $\{1\}$, $\{4\}$, it would find the optimum colouring $C_1 = \{2, 3, 5\}$ and $C_1 = \{1, 4\}$ (after all, the graph is bipartite). Intuitively, since the relative

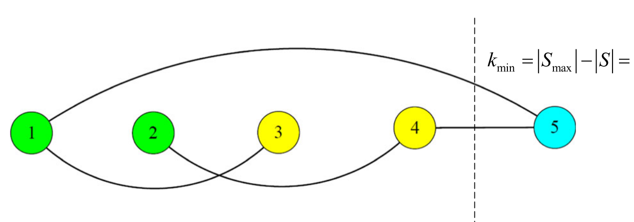



**Fig. 6** An example of a coloured graph

order of vertices determined initially remains the same for all subproblems (see Subsection 1.1), a colour-based sorting of vertices at the root node could improve the SEQ colourings of many subproblems (possibly also in the deeper levels of the search tree). This can prune the search space better (sometimes even exponentially better) than a standard degree-based ordering in some cases, as will be shown in the next section.

To summarize, we believe that COLOUR_SORT can be successful for the BBMC family of algorithms when the following two conditions are met:

- it is possible to greedily find a colouring of the input graph that is close to optimal.
- the chromatic number of the graph is a tight bound on its clique number.

Moreover, COLOUR_SORT can be even more effective if it is combined with a strong initial solution at the start of the search. As explained, a good initial lower bound would shift the threshold $k_{min}$ to the right and increase the number of colour classes to the left of the threshold in the shallow (and critical) levels of the search tree.

### 3.4 The initial sorting algorithm

Before selecting COLOUR_SORT as the initial sorting procedure, we first need to compare it with its degree-based counterpart. In [18], the *tail of the colouring*, that is, the colour classes with the highest colour numbers, is used for evaluation. A colouring is defined as *regular* if its tail contains not more than one colour class with a single vertex. If two or more singleton sets exist, it is considered irregular and dismissed.

In this work, we propose to compare any two initial vertex orderings for exact maximum clique search in the following manner. For a given vertex ordering $O = (v_1, v_2, \ldots, v_n)$, let $G_{v_1} = G[N_{\{v_1, v_2, \ldots, v_i-1\}}(v_i)]$ be the subproblem induced by the preceding neighbours of $v_1$ in the ordering and let $u(v_1) \geq 1+\omega(G_{v_i})$ be any upper bound on $\omega(G[N_{v_1,v_2,\ldots,v_i-1}(v_i) \cup v_1])$. We then define an upper bound for the ordering $O$ as $u(O) = \max_{v_i \in V}\{u(v_i)\}$. We consider the ordering $O_1$ to be preferable to the ordering $O_2$ if $u(O_1 < u(O_2))$.

With the help of this new bound $u(O)$, our algorithm NEW_SORT (Algorithm 2) evaluates both vertex ordering procedures —degree-based $O_{deg}$ (described in Section 2) and colour-based $O_{color}$— and selects the one with smallest value of $u(O)$. There are different ways to compute valid upper bounds for an ordering according to our previous definition. NEW_SORT uses greedy colouring SEQ (step 5). The notation $SEQ_{O_{deg}}$ indicates that $O_{deg}$ is the initial order of vertices for SEQ. A value of $u(O_{color})$ is equal to the number of colours of the RLF colouring $\{C_1, \ldots, C_k\}$ computed by COLOUR_SORT. This is because, in this colouring, $v_i \in C_j \Rightarrow u(v_i) = j$ for any vertex in the ordering. Based on the $u(o)$ value for both orderings, a decision is made; NEW_SORT selects $O_{color}$ if $k$ is strictly lower than $u(O_{deg})$ and selects $O_{deg}$ otherwise (step 6).

**Algorithm 2** The NEW_SORT algorithm

*Input*: A simple graph $G = (V, E)$
*Output*: An ordering of $V$
*Initial values*: $O_{deg} \leftarrow \phi$, $O_{color} \leftarrow \phi$, $k \leftarrow 0$

NEW_SORT(G)
1. $O_{deg} \leftarrow DEG\_SORT(G)$
2. **if** $p \leq 0.7$ **then return** $O_{deg}$ *//p is the uniform density of G*
3. **else**
4. $(k, O_{color}) \leftarrow COLOUR\_SORT(G)$
5. $u \leftarrow 1 + \max_{v_i \in V}$(number of colors required by $\text{SEO}_{O_{deg}}(G[N_{\{v_1,\ldots,v_{i-1}\}}]))$
6. **if** $(k < u)$ **return** $O_{color}$
7. **else return** $O_{deg}$
8. **endif**
9. **endif**

Finally, we note that if the input graph is not sufficiently dense, the task of finding a maximum clique in the complement graph becomes impractical. To avoid this, NEW_SORT follows the same strategy as [18] and dismisses $O_{color}$ if the average density of the graph $p(G)$ is below a certain threshold (step 2).

## 4 Experiments

The hardware used for the experiments was a 20 core Xeon with 128 Gb of RAM and Linux OS. All the algorithms considered were run on a single core. These were the following:

- BBMCX [15]: The most recent and efficient variant of the BBMC family of algorithms. Worth noting is the fact that in the comparison survey [21], the bitstring kernel of BBMCX [10–12] reported the best performance over a set of graphs from public benchmarks. A similar comment appears in a more recent survey [27], and therefore we consider the choice of BBMCX justified.
- MaxCLQ [14]: A state-of-the-art PMAX-SAT-based maximum clique solver, which uses an upper bound based on the Partial MAXimum SATisfiability problem. It was considered very efficient in [27].

For this report we consider the following initial sorting procedures:

**Table 3** Evaluation of NEW_SORT

| | | | MaxCLQ [14] | | BBMCX [15] | | | BBMCX + NEW_SORT | | | BBMCX/NEW | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\omega$ | $\omega_0$ | Steps | time | steps | time | $\omega_0$(ILS) | steps | time | p | steps | time |
| C125.9 | 34 | 32 | 1201 | 0.020 | 2919 | 0.025 | 34 | 1854 | 0.019 | 3 | 1.6 | 1.3 |
| C250.9 | 44 | 37 | 21226766 | 249 | 77522385 | 738 | 44 | 53944321 | 542 | 3 | 1.4 | 1.4 |
| MANN_a27 | 126 | 125 | 2006 | 0.130 | 4679 | 0.178 | 126 | 4552 | 0.182 | any | 1.0 | 1.0 |
| MANN_a45 | 345 | 342 | 70262 | 16.68 | 118384 | 37.43 | 344 | 117529 | 37.27 | any | 1.0 | 1.0 |
| MANN_a9 | 16 | 16 | 7 | <0.001 | 16 | <0.001 | 16 | 16 | <0.001 | any | 1.0 | 1.0 |
| brock200_1 | 21 | 17 | 35909 | 0.350 | 37153 | 0.183 | 21 | 28915 | 0.154 | 4 | 1.3 | 1.2 |
| brock200_2 | 12 | 9 | 1094 | 0.020 | 379 | 0.002 | 12 | 143 | 0.002 | 4 | 2.7 | 1.1 |
| brock200_3 | 15 | 11 | 2356 | 0.030 | 1543 | 0.009 | 15 | 1497 | 0.011 | 4 | 1.0 | 0.8 |
| brock200_4 | 17 | 14 | 10375 | 0.090 | 5556 | 0.035 | 17 | 2785 | 0.019 | 4 | 2.0 | 1.8 |
| brock400_1 | 27 | 21 | 18372917 | 185 | 16420935 | 143 | 25 | 12997259 | 118 | 4 | 1.3 | 1.2 |
| brock400_2 | 29 | 19 | 8169969 | 85.47 | 7024159 | 63.72 | 25 | 4101170 | 45.72 | 4 | 1.7 | 1.4 |
| brock400_3 | 31 | 20 | 15190268 | 144 | 13964442 | 110 | 31 | 914128 | 13.71 | 4 | 15.3 | 8.0 |
| brock400_4 | 33 | 20 | 9063733 | 89.94 | 7370261 | 63.56 | 33 | 374392 | 6.92 | 4 | 19.7 | 9.2 |
| brock800_1 | 23 | 17 | 359232471 | 4140 | 159657583 | 2233 | 21 | 154387948 | 2192 | 4 | 1.0 | 1.0 |
| brock800_2 | 24 | 16 | 293733416 | 3596 | 147005629 | 2038 | 21 | 120270391 | 1794 | 4 | 1.2 | 1.1 |
| brock800_3 | 25 | 16 | 173918786 | 2248 | 92850736 | 1324 | 22 | 59912902 | 1011 | 4 | 1.5 | 1.3 |
| brock800_4 | 26 | 17 | 125189650 | 1671 | 60470543 | 920 | 21 | 46774809 | 788 | 4 | 1.3 | 1.2 |
| dsjc1000.1 | 6 | 5 | 966 | 0.630 | 254 | 0.005 | 6 | 208 | 0.003 | 4 | 1.2 | 1.5 |
| dsjc1000.5 | 15 | 11 | 21048687 | 245 | 6245865 | 79.44 | 15 | 5807719 | 76.20 | 4 | 1.1 | 1.0 |
| dsjc500.1 | 5 | 4 | 439 | 0.100 | 21 | <0.001 | 5 | 5 | <0.001 | 4 | 4.2 | 1.0 |
| dsjc500.5 | 13 | 11 | 275859 | 2.62 | 117613 | 0.738 | 13 | 103363 | 0.631 | 4 | 1.1 | 1.2 |
| frb30-15-1 | 30 | 26 | 32046329 | 515 | 110874737 | 991 | 30 | 1 | <0.001 | 8 | 1.1E+08 | 9.9E+05 |
| frb30-15-2 | 30 | 24 | 42946247 | 696 | 74975855 | 669 | 30 | 1 | <0.001 | 8 | 7.5E+07 | 6.7E+05 |
| frb30-15-3 | 30 | 24 | 27267464 | 440 | 37741280 | 333 | 30 | 1 | <0.001 | 8 | 3.8E+07 | 3.3E+05 |
| frb30-15-4 | 30 | 25 | 55238114 | 879 | 121534953 | 1058 | 30 | 2 | <0.001 | 8 | 6.1E+07 | 1.1E+06 |
| frb30-15-5 | 30 | 24 | 40485579 | 652 | 83244519 | 721 | 30 | 2 | <0.001 | 8 | 4.2E+07 | 7.2E+05 |
| gen200_p0.9_44 | 44 | 33 | 6977 | 0.110 | 25638 | 0.208 | 44 | 5562 | 0.079 | 4 | 4.6 | 2.6 |
| gen200_p0.9_55 | 55 | 37 | 7576 | 0.110 | 45663 | 0.347 | 55 | 108 | 0.006 | 4 | 423 | 57.8 |
| gen400_p0.9_55 | 55 | 44 | – | >24h | 1834718567 | 24722 | 55 | 1 | <0.001 | 4 | 1.8E+09 | 2.5E+07 |
| gen400_p0.9_65 | 65 | 41 | 2038023795 | 33721 | – | >24h | 65 | 1 | <0.001 | 4 | $\infty$ | $\infty$ |
| gen400_p0.9_75 | 75 | 43 | 565148461 | 9182 | – | >24h | 75 | 373 | 0.012 | 4 | $\infty$ | $\infty$ |
| hamming10-2 | 512 | 512 | 1 | 0.760 | 1 | 0.018 | 512 | 1 | 0.024 | 4 | 1.0 | 0.8 |
| hamming8-2 | 128 | 128 | 1 | 0.010 | 1 | 0.001 | 128 | 1 | <0.001 | 4 | 1.0 | 1.0 |
| hamming8-4 | 16 | 16 | 1996 | 0.040 | 2905 | 0.015 | 16 | 2780 | 0.020 | 4 | 1.0 | 0.8 |
| johnson16-2-4 | 8 | 8 | 45738 | 0.470 | 102230 | 0.059 | 8 | 102230 | 0.059 | 4 | 1.0 | 1.0 |
| johnson8-2-4 | 4 | 4 | 9 | <0.001 | 9 | <0.001 | 4 | 8 | <0.001 | 4 | 1.1 | 1.0 |
| johnson8-4-4 | 14 | 14 | 6 | <0.001 | 40 | <0.001 | 14 | 29 | <0.001 | 4 | 1.4 | 1.0 |
| keller4 | 11 | 8 | 1348 | 0.020 | 2009 | 0.010 | 11 | 1485 | 0.010 | 4 | 1.4 | 1.0 |
| keller5 | 27 | 16 | 266122691 | 4505 | 3702995074 | 44109 | 27 | 119012711 | 1346 | 4 | 31.1 | 32.8 |
| p_hat1000-1 | 10 | 9 | 48774 | 1.53 | 20007 | 0.134 | 10 | 18567 | 0.148 | 10 | 1.1 | 0.9 |
| p_hat1000-2 | 46 | 40 | 4518975 | 90.36 | 2180084 | 57.82 | 46 | 983911 | 29.95 | 10 | 2.2 | 1.9 |
| p_hat1500-1 | 12 | 10 | 300447 | 9.22 | 106757 | 1.22 | 12 | 85739 | 1.14 | 10 | 1.2 | 1.1 |
| p_hat1500-2 | 65 | 58 | 229810475 | 7188 | 107564564 | 5260 | 65 | 37287206 | 2098 | 10 | 2.9 | 2.5 |
| p_hat300-1 | 8 | 7 | 424 | 0.040 | 236 | 0.001 | 8 | 169 | 0.002 | 10 | 1.4 | 0.5 |
| p_hat300-2 | 25 | 23 | 1041 | 0.040 | 470 | 0.012 | 25 | 196 | 0.006 | 10 | 2.4 | 2.1 |
| p_hat300-3 | 36 | 33 | 91552 | 1.08 | 59846 | 0.592 | 36 | 12004 | 0.164 | 10 | 5.0 | 3.6 |
| p_hat500-1 | 9 | 8 | 4387 | 0.160 | 629 | 0.008 | 9 | 599 | 0.007 | 10 | 1.1 | 1.2 |

**Table 3** (continued)

| | ω | $\omega_0$ | MaxCLQ [14] Steps | time | BBMCX [15] steps | time | $\omega_0$(ILS) | BBMCX + NEW_SORT steps | time | p | BBMCX/NEW steps | time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| p_hat500-2 | 36 | 33 | 22343 | 0.470 | 7451 | 0.109 | 36 | 1980 | 0.050 | 10 | 3.8 | 2.2 |
| p_hat500-3 | 50 | 44 | 2166987 | 36.44 | 1517820 | 30.63 | 50 | 644203 | 14.93 | 10 | 2.4 | 2.1 |
| p_hat700-1 | 11 | 8 | 14941 | 0.440 | 1649 | 0.022 | 11 | 631 | 0.017 | 10 | 2.6 | 1.3 |
| p_hat700-2 | 44 | 41 | 141935 | 2.76 | 58199 | 1.22 | 44 | 17493 | 0.465 | 10 | 3.3 | 2.6 |
| p_hat700-3 | 62 | 56 | 37780362 | 800 | 17326756 | 589 | 62 | 5971698 | 233 | 10 | 2.9 | 2.5 |
| san1000 | 15 | 8 | 21754 | 0.730 | 17395 | 0.531 | 8 | 17218 | 0.550 | 5 | 1.0 | 1.0 |
| san200_0.7_1 | 30 | 17 | 107 | 0.010 | 285 | 0.003 | 30 | 1 | <0.001 | 5 | 285 | 3.0 |
| san200_0.7_2 | 18 | 12 | 294 | 0.020 | 205 | 0.003 | 18 | 1 | <0.001 | 5 | 205 | 3.0 |
| san200_0.9_1 | 70 | 45 | 216 | 0.010 | 300 | 0.016 | 70 | 1 | <0.001 | 5 | 300 | 16.0 |
| san200_0.9_2 | 60 | 38 | 7087 | 0.100 | 31168 | 0.315 | 60 | 2 | <0.001 | 5 | 1.6E+04 | 315 |
| san200_0.9_3 | 44 | 31 | 11871 | 0.170 | 3328 | 0.028 | 44 | 5 | <0.001 | 5 | 666 | 28.0 |
| san400_0.5_1 | 13 | 7 | 457 | 0.050 | 542 | 0.006 | 13 | 1 | <0.001 | 5 | 542 | 6.0 |
| san400_0.7_1 | 40 | 21 | 4523 | 0.120 | 9495 | 0.151 | 40 | 1 | 0.003 | 5 | 9.5E+03 | 50.3 |
| san400_0.7_2 | 30 | 15 | 644 | 0.070 | 2971 | 0.039 | 30 | 1 | 0.002 | 5 | 3.0E+03 | 19.5 |
| san400_0.7_3 | 22 | 13 | 35049 | 0.410 | 54673 | 0.435 | 22 | 1 | 0.003 | 5 | 5.5E+04 | 145 |
| san400_0.9_1 | 100 | 48 | 19481 | 0.740 | 282092 | 4.44 | 100 | 1 | 0.003 | 5 | 2.8E+05 | 1480 |
| sanr200_0.7 | 18 | 14 | 20375 | 0.160 | 16428 | 0.077 | 18 | 11981 | 0.070 | 5 | 1.4 | 1.1 |
| sanr200_0.9 | 42 | 36 | 351501 | 3.94 | 855156 | 7.78 | 42 | 386841 | 3.90 | 5 | 2.2 | 2.0 |
| sanr400_0.5 | 13 | 12 | 69118 | 0.640 | 25349 | 0.141 | 13 | 15343 | 0.116 | 5 | 1.7 | 1.2 |
| sanr400_0.7 | 21 | 17 | 7866669 | 73.43 | 5967522 | 39.49 | 21 | 5834680 | 38.78 | 5 | 1.0 | 1.0 |

Times are measured in seconds with precision of milliseconds. For each row, the best time is shown in bold and the minimum number of steps in italics (a step is a call to the recursive search procedure). *BBMCX/NEW* columns report the ratio of time and steps between BBMCX without and with NEW_SORT. $\omega_0$ refers to the size of the initial solution computed by BBMCX (and also used by MaxCLQ). $\omega_0$ (ILS) refers to the size of the initial solution computed by the leading approximate local search heuristic ILS. Time cells with times below 1 ms (<0.001) are counted as 0.001 for the *BBMCX/NEW* ratio

- MW: Minimum width sorting of vertices.
- MWS: Minimum width sorting, breaking ties by minimum vertex support $\sigma$. In all graphs over (and including) 1,000 vertices, $\sigma$ has been computed statically (MWSS) because it is much faster.
- NEW_SORT: the sorting procedure described in Algorithm 2, which selects the best ordering between DEG_SORT and COLOUR_SORT. DEG_SORT is implemented with the parameter $p \in \{3, 4, \ldots, 10\}$ tuned for the best performance for each family of graphs. For this task we consider only easy instances in each family, that is, graphs with estimated running times below 5s. Thus, the tuning process does not constitute a significant constraint in practice.

We also compute a strong initial solution with a state-of-the-art heuristic. This was reported to improve the performance of exact maximum clique solvers in [17]. It was also discussed in Section 3.3 as a possible enhancement of COLOUR_SORT. The heuristic we used was ILS (Iterated Local Search, described in [28]) as in the original paper [17]. In all experiments, time is measured in seconds (with precision of milliseconds) and only running times for the actual search are given (the common procedure in maximum clique literature). The time limit for each experiment was fixed at 24 h.

Graphs employed for the tests are taken from DIMACS[1] (presented at the Second DIMACS Implementation Challenge) and BHOSHLIB[2] public data sets. The concrete 67 instances chosen are representative of all families and frequently used in similar reports that may be found elsewhere.

Table 3 reports all the results used to evaluate NEW_SORT. The best time for each graph is shown in bold and the minimum number of steps is shown in italics. The column header $\omega_o$ shows the initial clique computed during standard preprocessing. The column header $\omega_o$(ILS) shows the initial clique found by the ILS heuristic, which was optimal in 60 out of the 67 graphs considered. Concerning the algorithm configuration, MaxCLQ was run as provided by

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

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

the developer and given the same initial solution $\omega_0$ as the one computed by BBMCX; BBMCX + MW is the current release of BBMCX and BBMCX + NEW_SORT is the enhanced algorithm, which also includes the stronger $\omega_0$(ILS) lower bound. Finally, the column headers under BBMCX/NEW report the time and steps ratio between BBMCX + MW and BBMCX + NEW_SORT. In the cases where the performance of an algorithm is below a millisecond (reported as *<0.001*), the actual value is rounded up to a millisecond to compute the time ratio.

### 4.1 Evaluation

Of the 67 instances considered, BBMCX + NEW_SORT (or NEW_SORT for simplicity) performs better than BBMCX without NEW_SORT in 49 graphs. It is slower in only 5 graphs and prunes the search space better in 56 graphs. Moreover, the performance is improved by more than 15 times in 15 graphs, notably from the *gen*, *keller*, *frb*, and *san* families. Interestingly, NEW_SORT prefers COLOUR_SORT to the degree-based sorting computed by DEG_SORT in all graphs of three of those four families, specifically *gen*, *keller*, and *frb*.

We will now discuss the results for each family of graphs concerning BBMCX and NEW_SORT to try to provide explanations for the obtained results. The results by families may be summarized as follows:

– *MANN, hamming* and *johnson*: these sets are not significantly affected by any of the enhancements. A possible explanation for the *MANN* family is its very high density, which makes preprocessing irrelevant. The graphs from the other two families are easy for all the algorithms, so it is not possible to draw any conclusion.
– *C*: DEG_SORT, as well as the strong initial solution, explain the difference in performance of BBMCX for this family of graphs. We estimate the reduction of the search tree with the new initial ordering to be around 7 % in the more difficult *C250.9* graph.
– *brock* and *dsjc*: The impact of DEG_SORT is not very significant here. In the cases of almost an order of magnitude of improvement (i.e. *brock400_3* or *brock_400_4*), it is explained by a strong initial solution.
– *frb, gen*, and *keller*: When exponential improvements occur, the explanation is mainly due to COLOUR_SORT. Specifically, the *frb-30* instances have 30 as both the chromatic and the clique number, and DEG_SORT is unable to capture this structure. COLOUR_SORT, however, finds an optimum colouring, and when vertices are initially sorted in that way, the problem becomes trivial. Instances *gen400_p0.9_55* and *gen400_p0.9_65* are also trivially solved by BBMCX with COLOUR_SORT, while *keller5* is solved more than 30 times faster.
– *p_hat*: This family contains non-structured graphs in which significant differences between DEG_SORT and prior orderings were not expected. Interestingly, in three cases DEG_SORT reduces the size of the search tree by more than 10 %. Performances over this threshold are due to the improved initial solution.
– *san*, *sanr*: DEG_SORT improves performance by a small margin, compared with MW, in more difficult graphs (i.e. with 0.9 density). However, these types of instances are well known to be sensitive to a good solution, so whenever NEW_SORT gives a vast improvement in performance (as in the *san400_ 0.7—0.9* graphs), the main explanation is the strong initial solution.

Concerning parameter $p$ in DEG_SORT, the best overall value is 4 (in five families) followed by 5 (in *san* and *sanr*), 3 (in *C*), 8 (in *frb30*), and finally 10 for the *p_hat* family. As mentioned previously, the tuning procedure uses the easier instances, so it does not constitute a significant disadvantage in a real application.

With respect to MaxCLQ, the proposed NEW_SORT enhances BBMCX so that the latter performs better in the majority of graphs; specifically, it is faster in 60 cases, more than three times faster in 43 cases, and more than an order of magnitude faster in 26 cases. MaxCLQ is supposed to outperform standard BBMCX only in some of the harder, more dense, graphs (independently of the initial sorting). It does so significantly in the graphs *MANN_a27*,*MANN_a45*, and *C250.9*.

## 5 Conclusions

This work describes a new initial vertex ordering (NEW_SORT) that significantly improves the performance of a family of exact approximate-colour-based solvers for the MCP.

It does so by selecting the ”best” ordering between an improved typical degree-based ordering and a colour-based one. Both sorting procedures have polynomial time complexity and are easy to implement, which makes them useful in practical applications where the exact solution for the maximum clique problem is critical. The best results are obtained when NEW_SORT is further enhanced with a strong initial solution. The reported results show that the improved performance may even be exponential for some graphs.

As a side result, this work also provides an interesting observation for Erdös-Rényi uniform random graphs. It has been observed that the effectiveness of ordering vertices by non-increasing degree for sequential greedy colouring heuristic SEQ is inversely related to the size of these graphs.

Work in progress is concerned with further analysis of this result and, if considered appropriate, establishing theoretical proof.

**Acknowledgments** Pablo San Segundo and Alvaro Lopez are funded by the Spanish Ministry of Economy and Competitiveness (grants ARABOT: DPI 2010-21247-C02-01 and 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, NRU HSE. We would also like to thank Jorge Artieda for his help with the experiments. Finally, we express our gratitude to Chu-Min Li for providing the source code of MaxCLQ.

## Appendix

The list of instances from DIMACS and BHOSHLIB benchmarks employed in the reported results in Table 2 is:

*C125.9, C250.9, Mann_a9, Mann_a27, Mann_a45, brock200_1/4, brock_400_1/4, c-fat200-1, c-fat200-2, c-fat200-5, c-fat500-1, c-fat500-2, c-fat500-5, c-fat500-10, dsjc500.1, dsjc500.5, dsjc1000.1, dsjc1000.5, frb30-15-1/5, gen200_p0.9_44, gen200_p0.9_55, hamming6-2, hamming6-4, hamming8-2, hamming8-4, hamming10-2, johnons8-2-4, johnons8-4-4, johnons16-2-4, keller4, p_hat300-1/3, p_hat500-1/3, p_hat300-1/3, p_hat700-1/3, p_hat1000-1/2, p_hat1500-1, san200_0.7_1/2, san200_0.9_1/3, san400_0.5_1, san400_0.7_1/3, san400_0.9_1, san1000, sanr200_0.7, sanr200_0.9, sanr400_0.5, sanr400_0.9.*

## References

1. Konc J, Janezic D (2010) ProBiS algorithm for detection of structurally similar protein binding sites by local structural alignment. Bioinformatics 26:1160–1168
2. Eblen J, Phillips C, Rogers G, Langston M (2012) The maximum clique enumeration problem: algorithms, applications, and implementations. BMC Bioinforma 13:S5
3. Butenko S, Chaovalitwongse W, Pardalos P (eds) (2009) Clustering challenges in biological networks. World Scientific, Singapore
4. San Segundo P, Artieda J (2015) A novel clique formulation for the visual feature matching problem. Appl Intell 43(2):325–342
5. San Segundo P, Rodriguez-Losada D (2013) Robust global feature based data association with a sparse bit optimized maximum clique algorithm. IEEE Trans Robot 29(5):1332–1339
6. Östergård P (2002) A fast algorithm for the maximum clique problem. Discrete Appl Math 120:1:97–207
7. Fahle T (2002) Simple and fast: Improving a -and-bound algorithm for maximum clique. In: Proceedings ESA-2002, pp 485–498
8. 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
9. Tomita E, Sutani Y, Higashi T, Takahashi S, Wakatsuki M (2010) A simple and faster branch-and-bound algorithm for finding a maximum clique. LNCS 5942:191–203
10. San Segundo P, Rodriguez-Losada D, Jimenez A (2011) An exact bit-parallel algorithm for the maximum clique problem. Comput Oper Res 38:2:571–581
11. San Segundo P, Matia F, Rodriguez-Losada D, Hernando M (2013) An improved bit parallel exact maximum clique algorithm. Optim Lett 7:3:467–479
12. San Segundo P, Tapia C (2014) Relaxed approximate coloring in exact maximum clique search. Comput Oper Res 44:185–192
13. Li C-M, Quan Z (2010) An Efficient Branch-and-Bound Algorithm based on MaxSAT for the Maximum Clique Problem. In: Proceedings AAAI, pp 128–133
14. Li C-M, Quan Z (2010) Combining Graph Structure Exploitation and Propositional Reasoning for the Maximum Clique Problem. In: Proceedings ICTAI, pp 344–351
15. San Segundo P, Nikolaev A, Batsyn M (2015) Infra-chromatic bound for exact maximum clique search. Comput Oper Res 64:293–303
16. Konc J, Janečič D (2007) An improved branch and bound algorithm for the maximum clique problem. MATCH Commun Math Comput Chem 58:569–590
17. Batsyn M, Goldengorin B, Maslov E, Pardalos P (2014) Improvements to MCS algorithm for the maximum clique problem. J Comb Optim 27:397–416
18. Li C-M, Fang Z, Xu K (2013) Combining MaxSAT Reasoning and Incremental Upper Bound for the Maximum Clique Problem. In: Proceedings ICTAI, pp 939–946
19. Bron C, Kerbosch J (1973) Algorithm 457: finding all cliques of an undirected graph. Commun ACM 16:9:575–577
20. Balas E, Yu C (1986) Finding a maximum clique in an arbitrary graph. SIAM J Comput 15:4:1054–1068
21. Prosser P (2012) Exact algorithms for maximum clique: a computational study. Algorithms 5:4:545–587
22. Carraghan R, Pardalos P (1990) An exact algorithm for the maximum clique problem. Oper Res Lett 9:6:375–382
23. Personal communication with researchers Ciaran McCreesh and Patrick Prosser
24. Welsh D, Powell M (1976) An upper bound for the chromatic number of a graph and its application to timetabling problem. Comput J 10:1:85–86
25. Syslo M (1989) Sequential coloring versus Welsh-Powell bound. Discret Math 74:241–243
26. Leighton F (1979) A graph coloring algorithm for large scheduling problems. J Res Natl Bur Stand 84(6):489–506
27. Wu Q, Hao J (2015) A review on algorithms for maximum clique problems. Eur J Oper Res 242:3:693–709
28. Andrade D, Resende MG, Werneck R (2012) Fast local search for the maximum independent set problem. J Heuristics 18:4:525–547