On Maximum Weight Clique Algorithms, and How They Are Evaluated

Ciaran McCreesh, Patrick Prosser, Kyle Simpson, and James Trimble(✉)

University of Glasgow, Glasgow, Scotland
j.trimble.1@research.gla.ac.uk

Abstract. Maximum weight clique and maximum weight independent set solvers are often benchmarked using maximum clique problem instances, with weights allocated to vertices by taking the vertex number mod 200 plus 1. For constraint programming approaches, this rule has clear implications, favouring weight-based rather than degree-based heuristics. We show that similar implications hold for dedicated algorithms, and that additionally, weight distributions affect whether certain inference rules are cost-effective. We look at other families of benchmark instances for the maximum weight clique problem, coming from winner determination problems, graph colouring, and error-correcting codes, and introduce two new families of instances, based upon kidney exchange and the Research Excellence Framework. In each case the weights carry much more interesting structure, and do not in any way resemble the 200 rule. We make these instances available in the hopes of improving the quality of future experiments.

1 Introduction

This paper does not present a better algorithm for the maximum weight clique problem. Instead, it argues that due to questionable benchmarking practices, we do not know what the state of the art for maximum weight clique algorithms is. This is unfortunate, because the problem is widely researched.

Of particular interest to us is using a maximum weight clique algorithm to solve certain kinds of constraint optimisation problem. The reduction of constraint *satisfaction* problems to finding a clique in a corresponding *microstructure* graph is primarily studied for its theoretical properties [13–16, 28]. For problems with a special objective function, a reduction to the maximum clique problem which preserves the objective value is possible—indeed, recent experimental work shows that this encoding, rather than conventional constraint programming, is the best *practical* approach for solving the maximum common subgraph problem when vertex or edge labels are present [39]. To tackle other problems this way (such as graph edit distance problems with complex scoring schemes),

C. McCreesh, P. Prosser and J. Trimble—This work was supported by the Engineering and Physical Sciences Research Council [grant numbers EP/K503058/1, EP/M508056/1 and EP/P026842/1].

J.C. Beck (Ed.): CP 2017, LNCS 10416, pp. 206–225, 2017.
DOI: 10.1007/978-3-319-66158-2_14

we would like to be able to relax the restrictions on the objective function, by reducing to the maximum weight clique problem instead.

This paper also does not attempt to demonstrate that this is a viable approach. Although preliminary experiments suggest that this technique should not be dismissed out of hand, we believe that its chances of success would be much-improved by a change in how maximum weight clique algorithms are designed and evaluated. We therefore begin with a brief review of maximum weight clique algorithms. We then look at the set of instances usually used for benchmarking, focussing in particular upon a widespread practice of allocating weights to unweighted graphs. We show how this has affected the design of heuristics and other filtering rules. We finish by looking at other problem instances, where weights have real-world meanings and the vertices often have special structure; we make all of these instances available for other experimenters to use.

2 Maximum Weight Clique Algorithms

Given a graph G where each vertex v has an integer weight $\mathrm{w}(v)$, the *maximum weight clique problem* is to find a subset of vertices of maximum sum of weights, such that every vertex in the subset is adjacent to every other in the subset; note that the *maximum weight independent set* and *minimum weighted vertex cover* problems are essentially equivalent. The problem also comes in an edge-weighted variant, which we do not discuss in this paper. We write $\mathrm{N}(v)$ for the set of vertices adjacent to v (that is, its *neighbourhood*), the *degree* of a vertex is the cardinality of its neighbourhood, and the *density* of a graph is the proportion of potential edges which are present. We use C in our descriptions of algorithms to denote the current clique that is being built during search; P denotes the set of candidate (potential) vertices that may be added to this clique.

Cliquer [43] finds a maximum clique and an earlier paper [42] presents a sketch of how Cliquer can find a maximum weight clique. Cliquer is essentially a Russian Doll search [60], finding a maximum weight clique in iteration i with an initial clique $C = \{i\}$ and a candidate set of vertices to choose from $P = \mathrm{N}(i) \cap \{0, \dots, i-1\}$. The weight of this clique is then stored in an array element $c[i]$. In the case of unweighted maximum clique, $c[i] = c[i-1]$ if we cannot unseat the incumbent using vertices $\{0, \dots, i\}$. If we can unseat the incumbent using vertices $\{0, \dots, i\}$ then we must have added one more vertex, that vertex is i and $c[i] = c[i-1] + 1$. In the case of maximum weight clique, $c[i]$ is a weight, and $c[i] > c[i-1]$ if and only if we unseat the incumbent using vertices $\{0, \dots, i\}$. Obviously, $0 \leq c[i] - c[i-1] \leq \mathrm{w}(i)$. When a vertex v is selected from P to add to C, $c[v]$ can be used to prune the search. Prior to adding vertex v to C, we know that the best possible clique that can be found using vertices $\{0, \dots, v\}$ is $c[v]$, so if the weight of C plus $c[v]$ does not exceed the weight of the incumbent then search can be abandoned. The search is also pruned if the total weight of $C \cup P$ is no greater than the incumbent.

Kumlander's algorithm [32] is an enhancement of Cliquer. At the top of search the vertices of the graph are coloured, giving colour classes C_1 to C_k. In each colour class, vertices are then sorted by weight in ascending order. We now have a Cliquer-like search with iterations 1 to k, where iteration i finds the heaviest clique using vertices in the colour classes $C_1 \cup \ldots \cup C_i$ and the weight is recorded in $c[i]$ (as in Cliquer). This can then be used as a bound (as before) along with a *colouring upper bound*, that is the sum of the maximum weights in each of the colour classes $C_1 \ldots C_i$, i.e. $\sum_{j=1}^{i} \max\{\mathrm{w}(v) : v \in C_j \cap P\}$.

MWCLQ [22] uses MaxSAT reasoning to tighten an upper bound given by vertex colouring. At each search node, the vertices of G are first partitioned into independent sets $C_1 \cup \ldots \cup C_n$. This allows conversion to a literal-weighted MaxSAT encoding: a variable x_i is created for every vertex v_i, having $\mathrm{w}(x_i) = \mathrm{w}(v_i)$, and a hard clause $\overline{x}_i \vee \overline{x}_j$ is posted for each pair of vertices v_i, v_j which are not adjacent. For each independent set $C_i = \{v_1, \ldots, v_l\}$, a soft clause is then added where literals are weighted, $c_i = (x_1, \mathrm{w}(x_1)) \vee \cdots \vee (x_l, \mathrm{w}(x_l))$, where $\mathrm{w}(c)$ is the maximum literal weight within that clause and literals in a clause are sorted by weight in non-increasing order. The upper bound starts as the sum of all original soft clause weights. By applying unit propagation on an instance, the algorithm identifies sets S of conflicting soft clauses and for each, the accompanying set S_{topk} where the k highest-weight literals are failed. Defining $\mathrm{w}(S) = \min\{\mathrm{w}(c) : c \in S\}$ and $k(t)$ as the count of failed high-weight literals in t, the upper bound is then reduced by $\min(\mathrm{w}(S), \min_{t \in S_{topk}}(\mathrm{w}(t) - \mathrm{w}(x_{k(t)+1})))$. Further such sets (and bounds reductions) are identified by iteratively splitting and transforming the soft clauses in S and S_{topk} to obtain new unit clauses.

Tavares [2,59] introduces a new heuristic colouring algorithm for calculating an upper bound, *BITCOLOR*, in which each vertex may appear in more than one colour class. Each colour class has an associated weight, and the colouring has the property that the weight of a vertex v equals the sum of the weights of the colour classes to which v belongs. If we produce a colouring of the candidate set P in this way, and let $\mathrm{UB}(P)$ be the sum of colour-class weights, it can be shown that $\mathrm{UB}(P) + \sum_{v \in C} \mathrm{w}(v)$ is a valid upper bound on the maximum clique weight. In practice, this approach produces a tighter upper bound than simple colouring. Tavares uses BITCOLOR in three algorithms for maximum weight clique: a Cliquer-style Russian dolls algorithm, an algorithm which branches on vertices in reverse colouring order, and a resolution search algorithm [12].

OTClique [55] enhances the Russian dolls approach of Cliquer by precomputing, using dynamic programming, the maximum-weight clique in each of a large set of induced subgraphs of G. The precomputed values are used to quickly calculate a bound that is tighter than the naïve sum-of-vertex-weights bound used by Cliquer.

WLMC is an exact algorithm which is designed for large, sparse graphs, but also performs well on the relatively small, dense graphs that are the focus of this paper. In a preprocessing step—which is performed at the top of search and also after every possible choice of first vertex—WLMC uses the method of Carraghan and Pardalos [11] to produce a vertex ordering and an initial incumbent clique. The preprocessing step then reduces the size of the graph by deleting any vertex v such that the incumbent clique has weight greater than or equal to $\mathrm{w}(v)$ plus the sum of weights of v's neighbours. At each node of the branch-and-bound search, WLMC uses MaxSAT reasoning to find a set of vertices on which it is unnecessary to branch[1].

Other approaches. The problem has also been tackled using mathematical programming [25,27,61], and is the subject of ongoing research for inexact (heuristic) solutions [3,4,10,20,27,29,41,62,65,66]. Finally, sometimes alternative constraints or objectives are considered [6,34,54].

3 Current Practices in Benchmarking

For the maximum (unweighted) clique problem, experimenters are blessed with a suite of instances from the second DIMACS implementation challenge [30]. These are all relatively small, dense graphs. Most instances fit into one of three classes:

– Graphs which encode a problem from another domain. The “c-fat” family encode a problem involving fault diagnosis for distributed systems [5]. The “hamming” and “johnson” graphs model problems from coding theory [7]. The “keller” instances encode a geometric conjecture [17], and the MANN family is made from clique formulations of the Steiner triple problem [36]. In each of these cases, the size of the solution has a real-world interpretation (and sometimes the vertices contained therein also convey meaning).
– Randomly-generated graphs. The “C” and “DSJC” instances are simple random graphs of varying orders and densities. The “p_hat” family are also random graphs, but with an unusually large degree spread [23,56].
– Random graphs with large hidden solutions. The “brock” family of instances [9] are an attempt at camouflaging a known clique in a quasi-random graph for cryptographic purposes, in a way resistant to heuristic attacks. The “gen” and “san(r)” instances use a different technique for hiding a large clique of known size in a graph [50,51]: again, they are an attempt to create challenging instances with a known and unusually large optimal solution.

The instances from the first set are valuable because of their applications. Meanwhile, the randomly generated instances are useful because they provide a challenge: although being able to solve crafted hard instances is not the primary goal of developing clique algorithms, working on these instances has led to better solvers. For example, Depolli *et al.* [18] use a maximum clique algorithm

[1] The existing implementation of WLMC does not support the large weights that appear in many of the instances in this paper. Therefore, we could not include this program in our experimental evaluation.

for new instances from a biochemistry application, and note that although their instances are reasonably easy for a modern algorithm, they are challenging for earlier algorithms that predate experiments on these instances; a similar conclusion holds for clique-based solvers for maximum common subgraph problems [39].

For the weighted problem, standard practice is to take these same instances, and to follow a convention usually ascribed to Pullan [45]:

> "Instances were converted to MVWC instances (the DIMACS-VW benchmark) by allocating weight, for vertex i, of $i \bmod 200 + 1$".

Incidentally, Mannino and Stefanutti [37] had used a similar convention previously, using modulo 10 rather than 200. Pullan justifies this rule and choice of constant as follows:

> "This technique allows future investigators to simply replicate the experiments performed in this study. The constant 200 in the weight calculation was determined after a number of experiments showed that the generated problems appeared to be reasonably difficult for PLS (clearly, allocating weights in the range $1, \dots, k$ results in an MC instance when $k = 1$ while, intuitively, it is reasonable to expect that as k increases, the difficulty in solving the instance will, in general, increase)."

This rule, together with a similar rule for allocating weights to edges for the edge-weighted variant of the problem, is very widely used [2–4,20–22,25,27,29,32,34,41–43,45,55,61,62,65,66, and many more], often as the only way of evaluating a solver. It has also recently been adopted for large sparse graphs [10,20,29,62], and for benchmarking the minimum weight dominating set problem [63], often as the only way of evaluating a solver. It has also recently been adopted for large sparse graphs [10,20,29,62], and for benchmarking the minimum weight dominating set problem [63].

4 Experimental Setup

Our experiments are performed on a machine with dual Intel Xeon E5-2697A v4 CPUs and 512 GBytes RAM, running Ubuntu 16.04. We used the latest version of Cliquer (1.21), released in 2010, downloaded from the author's website. The source code for MWCLQ and OTClique was provided by these programs' authors. We modified each program by changing every occurrence of `int` to a 64-bit integer type, in order to accommodate the large weights that occur in some classes of instance. This change has a measurable effect on the runtime of the programs, particularly for OTClique.

Tavares' programs were not available when we ran our experiments. We have therefore written two programs using a Tavares-style colouring for use in our experiments, one which uses Russian Dolls and one which branches in an order based on colouring. We call these programs TR and TC, respectively. In both

Tavares' description and our implementations, bitsets are used to perform the colouring step efficiently.

All five programs are implemented in C, and were compiled using GCC 5.4.0 at optimisation level `-O3`. We set the OTClique parameter l to 20. Finally, we implemented a constraint programming model in Java, using the Choco solver version 3.3.3.

5 Does Weight Allocation Affect Algorithm Design?

The maximum weight clique problem has an obvious constraint programming model: we have a 0/1 variable for each vertex, a constraint for each non-adjacent pair of vertices prohibiting the two vertices from being set to 1 simultaneously, and an objective to maximise the weighted sum over all variables. But what about variable-ordering heuristics? For the unweighted maximum clique problem, we might use the degree of the vertex corresponding to each variable. In the weighted case, we could look either at degree, or at weight.

When weights are chosen to be between 1 and 200, we would expect weights to be much more important than degree: selecting a vertex of high weight would affect the solution more than selecting several vertices of low weight. Thus it seems likely that a variable-ordering heuristic which considered weights would be best. On the other hand, if we selected weights to be between 190 and 200, perhaps degree would matter more instead?

Figure 1 confirms this suspicion. We look at random graphs with 70 vertices and density 0.6. We assign weights sequentially, starting at $x + 1$ and wrapping back to $x + 1$ when we exceed 200. Thus, on the far left of the plot, we have weights ranging from 1 to 70, in the middle from 101 to 170, near the right from 180 to 200 (with weights repeated), and on the far right, every weight is 200. For the y-axis, we plot the mean search effort from a sample of 100 runs for our Choco model, using ascending and descending degree or weight as static variable-ordering heuristics, and preferring 1 over 0 as a value-ordering heuristic. Because weights are allocated sequentially, we randomly permute the order of vertices before solving to avoid using weight unintentionally as a tie-breaking heuristic. The results show that when weights are chosen to be between 1 and 70, it is indeed best to select a weight-based variable ordering heuristic. However, once weights are chosen to be between 50 and 119, it becomes much more useful to use degree-based heuristics.

The plot also shows the effects of using impact-based search [46]. These results are nearly as good as the degree-based heuristics, but do not beat tailored heuristics. Impact-based search is also unable to mimic weight-based heuristics in the parts of the parameter-space where weights are more informative, since impact is unaware of the effect of domain deletions upon the objective function. We also plot domain over weighted degree [8], which fares less well.

What about other densities? Most of the DIMACS instances are dense—does this affect algorithm design too? In the top left plot of Fig. 2, we vary both the graph density and weight range, and use colour to show which heuristic has best

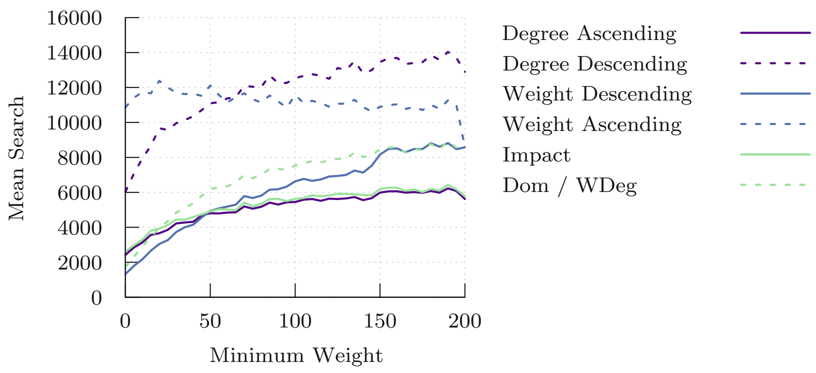

Fig. 1. A comparison of heuristics using a Choco model, over random graphs with 70 vertices, density 0.6, and sequential weights starting from x and never exceeding 200.

average performance at each point. The results show that when the minimum weight is low (such as when using the 200 rule), we should favour descending weight heuristics, but otherwise we should favour ascending degree.

This basic constraint programming approach is not performance-competitive, but the relative simplicity of the algorithm makes it easy to experiment with. What about the algorithms introduced in Sect. 2, which use more complex branching strategies and inference? Figure 2 also compares the run times of five dedicated algorithms with modified vertex orderings, working with 100 vertex graphs. (Note that several of the algorithms have branching strategies that are influenced by, but not identical to, the order of vertices in the graph.) The focus of this paper is not on explaining these results in great detail: we are simply demonstrating that the 200 rule has likely had an effect on algorithm design[2]. The default orderings for most of these algorithms are primarily weight-based, which appears to be a good choice when weight ranges are large (there are many dark and light blue points towards the bottom of these plots), but perhaps not otherwise (the plots are not monochromatic, and there are large orange and/or yellow areas in each plot).

Are heuristics the only design choice affected by the benchmark instances? Figure 3 compares our Tavares-style algorithm TC with a similar algorithm which uses a simple colour bound rather than the Tavares-style multi-colouring. The plots show mean search effort and runtimes for random 200-vertex instances of density 0.6. On the instances with a wide weight range such as 5–200, the

[2] It is interesting to note that MWCLQ often resembles Choco but with a higher threshold for switching away from weights, except that sometimes it is worth switching to descending degree as well as ascending degree, and that the three Russian Dolls algorithms exhibit similar heuristic behaviour to each other. We do not understand how ordering heuristics should work with Russian Dolls search, and suggest that this could be a good avenue for future research—for example, perhaps it is better to use different heuristics for different dolls?.

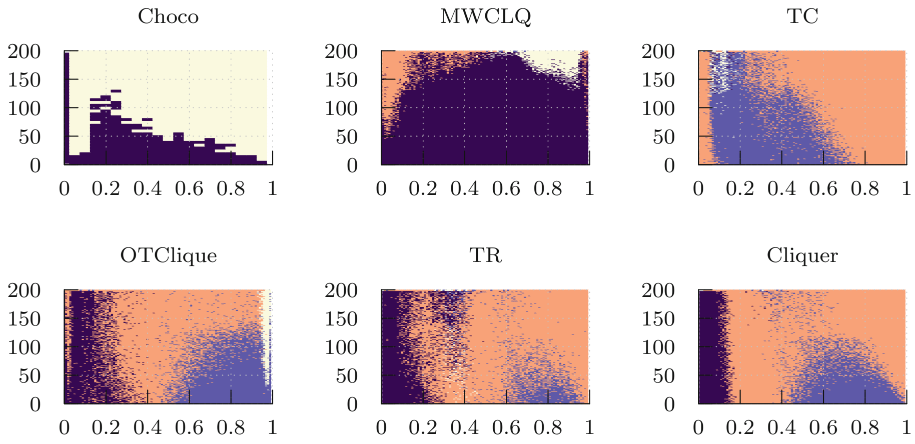

Fig. 2. Which heuristic is best when varying density (x-axis) and weight range (y-axis), choosing from descending/ascending weight (dark/medium blue), descending/ascending degree (orange/yellow), or no difference (white). Graphs have 70 vertices for Choco, and 100 for the other algorithms (which also have higher plotting densities). No plot is monochromatic, showing that heuristics are affected by density and weight ranges. (Color figure online)

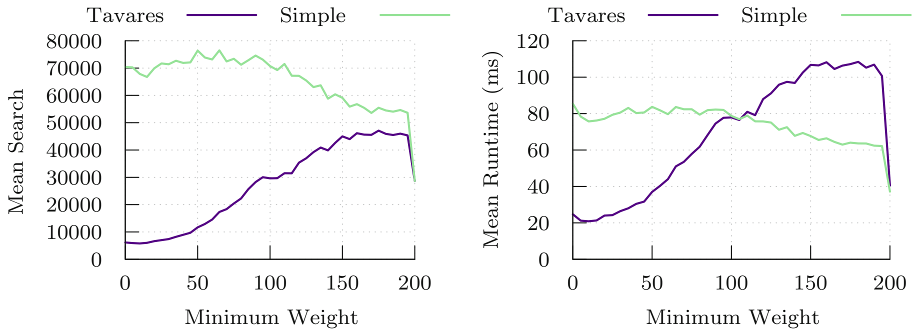

Fig. 3. Comparing TC with a simpler algorithm which does not use the Tavares multi-colouring rule, on 200 vertex graphs with density 0.6, and different weight ranges.

Tavares-style algorithm is the clear winner. When the minimum weight is greater than 100, the simple algorithm is faster; although it visits more search nodes, this is outweighed by the shorter time per node of the simpler algorithm. This shows that the practical benefits of Tavares' more complex bound are also heavily dependent upon how weights are allocated.

6 Other Families of Problem Instances

Having questioned the 200 rule and the use of unweighted DIMACS instances, we now discuss five families of instances which we hope will lead to better

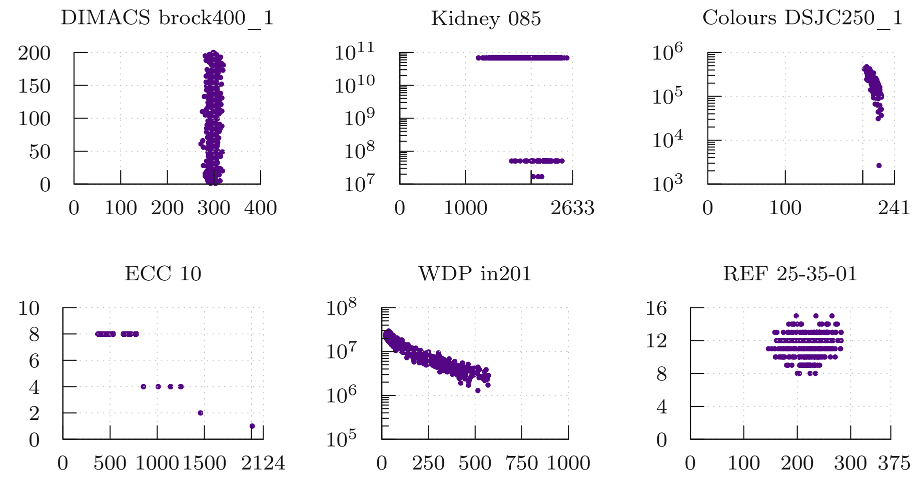

Fig. 4. Weight (y-axis) plotted against degree (x-axis) for an example instance from each different graph class. Note some plots use a log scale for the weight, and do not start at zero.

experiments in the future. Three of these are from existing papers (but in one case the instances are hard to find online in a convenient format), and two are new. We bring all of these instances together in the standard DIMACS format to help future experimenters, and we will update this collection as new families are uncovered[3]. Note that many of these instances require support for 64-bit weights.

Figure 4 plots weight versus degree for one instance from each of these families. We also plot "brock400_1" from the DIMACS set using the 200 rule: observe that degree gives almost no information for this instance, compared to weight. We return to this figure as we introduce each family.

6.1 Kidney Exchange

Kidney-exchange schemes exist in several countries to increase the number of transplants from living donors to patients with end-stage renal disease [24,35,47]. A patient enters the scheme along with a friend or family member who is willing to donate to that patient but unable to do so due to blood or tissue incompatibility. These two participants form a *donor-patient pair*. From the pool of donor-patient pairs, the scheme administrator periodically arranges *exchanges*, each of which involves two or more donor-patient pairs. In a two-way exchange, the donor of the first pair gives a kidney to the patient of the second pair, and the donor of the second pair gives a kidney to the patient of the first. In three-way and larger exchanges, kidneys are donated between the donor-patient pairs cyclically.

[3] https://doi.org/10.5281/zenodo.816293.

In addition, many schemes benefit from *altruistic donors*, who enter the scheme without a paired patient, and may initiate a chain of donations. For the optimisation problem, we may view an altruistic donor as a donor paired with a "dummy patient" who is compatible with any donor.

Each feasible exchange is given a score reflecting its desirability. This may, for example, take into account the size of the exchange, the time that patients have been waiting for a transplant, and the probability that the transplants will be successful. Typically, the scheme administrator carries out a *matching run* at fixed intervals, with the goal of maximising the sum of exchange scores. A popular approach to solving this optimisation problem is integer programming using the *cycle formulation*, in which we have one binary variable for each feasible exchange, and a constraint for each participant in the scheme ensuring that he or she is involved in at most one selected exchange [1,48]. We propose that this optimisation problem may, alternatively, be solved by reduction to maximum-weight clique. Each vertex is an exchange, whose weight is its score. Two exchanges are adjacent if and only if they have no participants in common.

To create maximum weight clique instances, we used kidney instances by Dickerson [19], available on PrefLib [38], originally from a widely-used generator due to Saidman *et al.* [49] (real instances cannot be made public due to medical confidentiality). The weighting scheme and exchange size cap we used are based on the system used in the UK's National Living Donor Kidney Sharing Scheme (NLDKSS) [35]. The NLDKSS has a maximum exchange size of three, and has five objectives, ranked hierarchically. The first objective is optimised; subject to this being at its optimal value, the second objective is optimised, and so on.

The primary objective is to maximise the number of *effective two-way exchanges*: exchanges that either consist of only two donor-patient pairs, or which contain (as part of a larger exchange) two donor-patient pairs who could by themselves form a two-way exchange. This provides robustness: part of a larger exchange may still proceed even if the full exchange does not (for example, due to illness). The second objective is to maximise the total number of transplants. The third objective is to minimise the number of three-way exchanges. The fourth objective is to maximise the number of *back-arcs* in three-way exchanges; these are compatibilities between donor-patient pairs in the reverse direction of the exchange. The final objective is to optimise the *weight* of the exchange, which is a value based on factors including the number of previous matching runs that patients have been in, and the level of compatibility between donors and patients in planned transplants.

To create these instances, we used the first four of these objectives, combining them into a single long integer using the formula $x = 2^{36}x_1 + 2^{24}x_2 + 2^{12}x_3 + x_4$ where x_i is an exchange's score for the ith objective. We use a simple transformation to convert the third objective from a minimisation to a maximisation. This method of combining scores in order to perform a single optimisation is not practical using IP solvers because, as Manlove and O'Malley [35] observe, the resulting weights would be too large for IP solvers. By contrast, all of our maximum-weight clique solvers can use 64-bit weights without loss of precision.

(Ideally, we would like to use even larger weights, to include the fifth ranking criterion.) Note that due to the extreme ranges of weights requiring the use of a log scale, Fig. 4 does not clearly show the variation between weights.

6.2 Colouring Instances

In branch and bound graph colouring algorithms such as Held *et al.* [26] the *fractional chromatic number* $\chi_f(G)$ acts as a useful upper bound. This can be found according to an integer programming formulation introduced by Mehrotra and Trick [40]: the model starts with a subset of the required variables, which is extended if a *maximum weight independent set* (MWIS) of weight at least 1 can be found within the original graph. The weights themselves are the *dual price* of including that vertex in the model according to an independent set formulation, multiplied by some factor *scalef* to achieve integer values. As a result, these graphs feature very large weights to have sufficient resolution to encode small fractions of *scalef*.

The instances we include are due to Held *et al.* Each instance arises during colouring of a corresponding DIMACS instance; many of these are the last such MWIS instance encountered during search, and represent that problem's bottleneck.

6.3 Error-Correcting Codes

Östergård [42] describes the following problem from coding theory. Let a length n, a distance d, a weight w, and a permutation group G be given. The objective is to find a maximum-cardinality set C of codes (binary vectors) of length n, such that each element in S has Hamming weight w; each pair of elements in C is at least Hamming distance d apart; and for every permutation $\sigma \in G$ and every code $c \in C$, we have that $\sigma(c) \in C$. Östergård shows how this problem may be reduced to maximum weight clique by partitioning the set of all binary vectors of length n and weight w into orbits under the permutation group G, and creating a vertex for each orbit satisfying the condition that no two members of the orbit are less than Hamming distance d apart. The weight of each vertex equals the size of the corresponding orbit, and two vertices are adjacent if and only if all pairs of members of the two orbits are at least distance d apart.

The fifteen instances presented by Östergård are no longer readily available. We have written a program to reconstruct the instances. For the instance ECC10 shown in Fig. 4 the weights range from one to eight, and are roughly inversely correlated with degree; in other instances, the weights go as high as eighty.

6.4 The Winner Determination Problem

In a combinatorial auction, bidders are allowed to bid on sets of items rather than just single items. For example, at a furniture auction, agent A might bid for four dining chairs and a table, rather than bid for each chair and the table separately. Another bidder, agent B, might bid for the same table and a sideboard, whilst agent C bids for the sideboard and a set of crockery. Agent B's bid is incompatible with that of A (they want the same table) and that of C (they want the same sideboard), but A's bid is compatible with C's (there is no intersection on the items of interest).

Finding an allocation of items to bidders that maximizes the auctioneer's revenue is called the winner determination problem (WDP) [44,52,53]. A problem instance can be represented as a weighted graph. A vertex v in the graph corresponds to a bid, the weight of v is the value of that bid, and an edge exists between a pair of vertices (u, v) if the corresponding bids have no items in common (i.e. they are compatible with each other). Consequently, a maximum weight clique corresponds to an optimal allocation.

WDP instances, available via cspLib [44] and originally created by Lau and Goh [33], have been used as a benchmark suite by Fang *et al.* [22] and Wu and Hao [64] for comparing one maximum weight clique algorithm against another. But what do these instances look like? Figure 4, instance WDP in201, shows that high weight vertices have low degree, and light weight vertices have high degree. This is not surprising: a high value bid is typically a bid for many items and is incompatible with many other bids, and corresponds to a heavy vertex of low degree. Consequently, when used to compare algorithms, we might find that an ordering on decreasing weight will perform much the same as an ordering of increasing degree.

6.5 The Research Excellence Framework

In 2016, Her Majesty's Government proposed that in the next Research Excellence Framework (REF2021) academics would be allowed to submit exactly four publications over a given interval of time (typically 4 years)[4]. In a university, in each unit of assessment (typically a department or school) each member of staff would submit six publications and of those six publications management would select four. Papers are assigned rankings in the range 4 to 1, with 4 being "internationally excellent". Therefore, for each member of staff, there would be C_4^6 possible selections, where each selection would have a combined ranking in the range 4 to 16. At most one of these 4-selections would be allowed for each member of staff, and no publication could be counted more than once (that is, co-authors within the same unit of assessment cannot both submit a shared publication).

This has strong similarities to a winner determination problem: we must find an allocation of items (sets of four publications) to bidders (academic staff) that maximizes the auctioneer's (unit of assessment's) revenue (combined rankings).

Realistic instances were generated for departments with n members of academic staff producing m publications. A random number of papers were generated, each with a ranking in the range 2 to 4, with a specified distribution based on historical data[5]. For each member of staff 6 papers were randomly selected, and

[4] However, in July 2016 Lord Nicholas Stern suggested greater flexibility be allowed.

[5] Being a *"research-led institution"* no papers with a ranking of 1 are allowed.

that member of staff was then considered an author. This was then represented as a weighted graph. The graph has $15 \cdot n$ vertices (there are 15 ways for each author to choose 4 publications from 6) with weights in the range 8 to 16. The 15 vertices associated with an author form an independent set (at most one of the author's 4-selections can be selected).

As the number of publications to choose from increases, the likelihood of any pair of 4-selections having a publication in common falls, so bids become more compatible and the resultant graph has more edges (is denser), and this tends to increase the difficulty of the problem. For example with $n = 20$ and $m = 50$ graphs have 300 vertices and average density 0.67, and with $n = 20$ and $m = 30$ we again have 300 vertices and density is 0.52 on average. These graphs have a maximum (unweighted) clique of size no more than $\min(n, m/4)$. There is a small range of weights (8 to 16) and in any instance there is a small variation in degree (see instance REF 25-35-01 in Fig. 4).

6.6 Experiments

In Figs. 5, 6, 7, 8, 9 and 10 we plot, for each algorithm, the cumulative number of instances which can be solved in under a certain amount of time, for these different families of instances. The dark thick line in each plot shows that algorithm's default vertex ordering, and the light lines show ascending and descending weight and degree orderings. To interpret these plots, select a preferred timeout along the x-axis, and then select the line with highest y-value to determine the best-performing algorithm for that choice of timeout.

Although we did not intend to carry out an algorithmic beauty contest, these results support the simple conclusion that our implementation of Tavares' (little known) colour-ordering algorithm is consistently the best solver, and that the default heuristic we picked for it (decreasing degree order) is nearly always the best. This is a surprise. We were hoping to end this paper by stressing the importance of tailoring heuristics and solvers on a family by family basis, perhaps suggesting algorithm portfolios, but instead we have identified a clear winner.

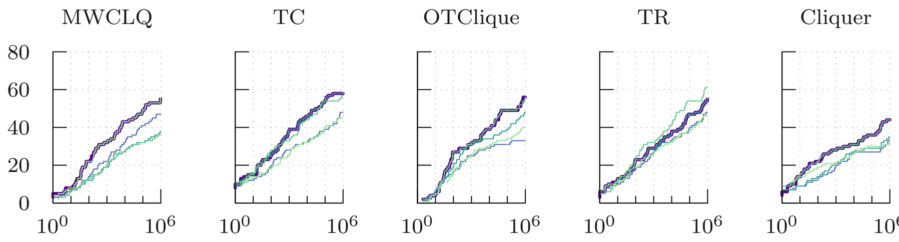

Fig. 5. Cumulative plots for DIMACS instances, with weights in range 1–200 added using the standard scheme. The dark thick line is the default heuristic for each solver, and the thin light lines show ascending and descending degree and weight heuristics. The x-axis is runtime in milliseconds, and the y-axis plots the cumulative number of instances which can be solved (individually) in time less than or equal to x.

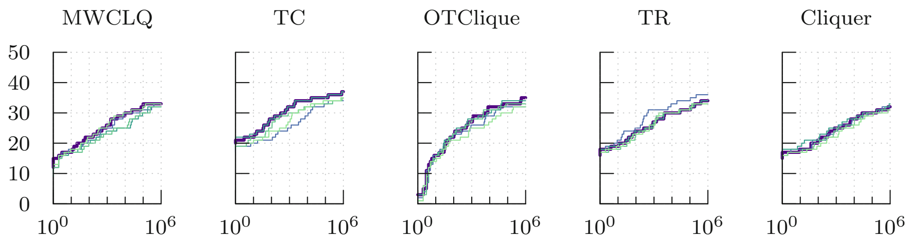

Fig. 6. Cumulative plots for kidney instances.

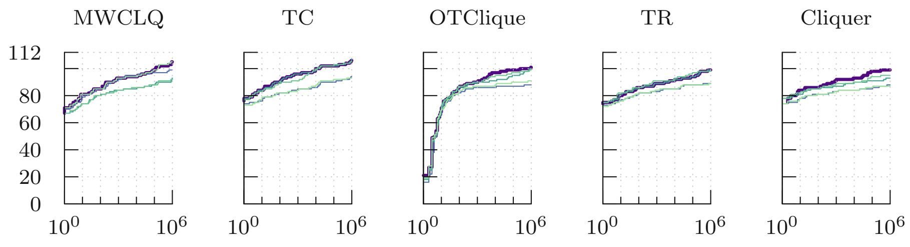

Fig. 7. Cumulative plots for colouring instances.

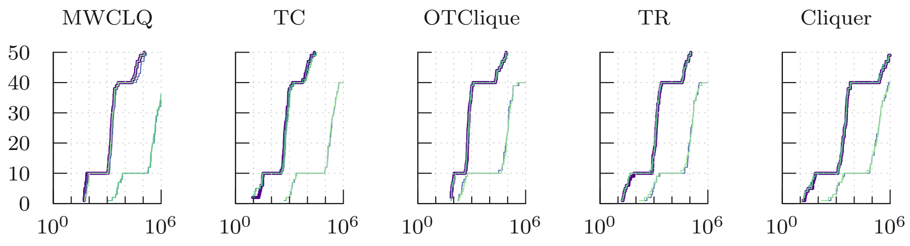

Fig. 8. Cumulative plots for winner determination problem instances.

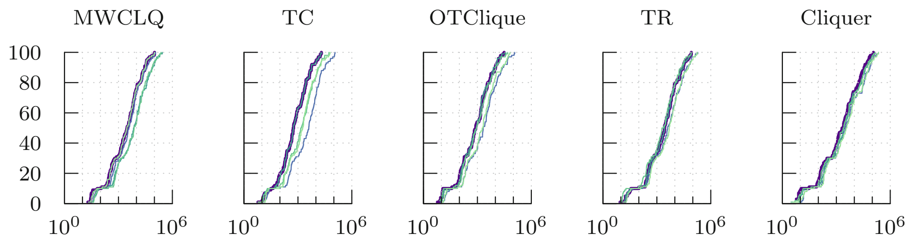

Fig. 9. Cumulative plots for REF instances.

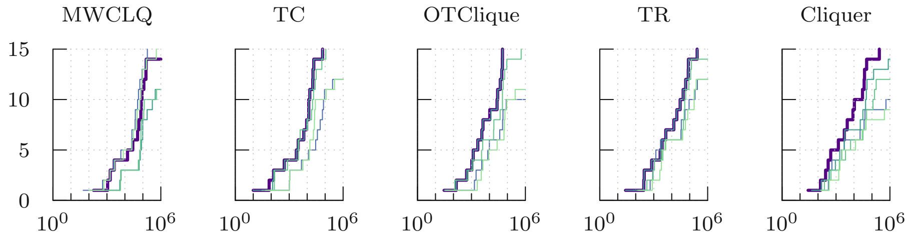

Fig. 10. Cumulative plots for error correcting code instances.

7 Conclusion

Despite our experiments suggesting a single winning algorithm, we believe our new sets of instances are valuable. The 200-weighted DIMACS benchmark instances are often cited as being good "real-world" tests for the maximum weight clique problem. This is not the case: some of these instances are real-world tests for the maximum clique problem, but adding weights destroys the real-world meaning of the results. Additionally, most of these instances are of the "crafted, challenging" (for unweighted clique) kind, and again, adding 200-rule weights destroys these properties. The other families we discuss in this paper are somewhat better in this respect, and if they replace the 200-weighted DIMACS instances as the standard for benchmarking, they may open the way up more interesting kinds of algorithm in the future.

Figure 6 emphasises this opportunity. It shows 50 kidney-exchange instances which have a minimum of 16 pairs and no altruistic donors, and a maximum of 64 pairs and 6 altruistic donors. These results are far from competitive with leading integer program solvers, which can solve each of these instances in less than a second.

Our discussion has focussed on weights. However, it is worth noting that for many of the DIMACS instances, vertex degrees are also unusually unhelpful. The situation shown in the top left plot of Fig. 4 where each vertex has similar degree is common, and for some instance families, the degrees are deliberately constructed to be misleading. In contrast, the vertices in our instances were not crafted with hostile intent, and they often carry a certain amount of structure. This is particularly true with microstructure-like encodings, where vertices from any given variable always form an independent set, and where we know that the graph may always be coloured in a particular way. Now that we have families of instances that have interesting, realistic structure, perhaps subsequent algorithms can be tailored to exploit these properties (such as treating the first branching vertex specially [58]), and it may also be worth considering preprocessing techniques [57].

We hope to extend our collections of instances and algorithms in the future, and perhaps this will make these results more interesting and inspiring. We are also interested in real instances for the edge-weighted variant of the problem, which suffers similarly from an arbitrary weight allocation rule.

We note in passing that all of these instances are dense, despite being “real-world” instances. It is important to distinguish between solving graph problems on graphs which directly *represent* real-world phenomena (which are often sparse and have power-law degree structures), with solving problems which *encode* the solution to a problem. Graphs of the latter kind may very well be dense. This is true even when the question being answered is regarding a sparse graph: for example, when solving the maximum common subgraph problem via reduction to clique, the encoding of two sparse graphs gives a dense graph [39]. Similarly, microstructure graphs for non-trivial problems are usually reasonably dense.

Finally, we observed (in Fig. 2) an anomaly with respect to the variable-ordering heuristics used by Russian Doll algorithms. Clearly, this deserves more attention.

Acknowledgments. The REF instance generator was joint work with David Manlove. We are grateful to David for this, and for helpful discussions on kidney exchange.

References

1. Abraham, D.J., Blum, A., Sandholm, T.: Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. In: MacKie-Mason, J.K., Parkes, D.C., Resnick, P. (eds.) Proceedings 8th ACM Conference on Electronic Commerce (EC-2007), 11–15 June 2007, San Diego, California, USA, pp. 295–304. ACM (2007). http://doi.acm.org/10.1145/1250910.1250954
2. Araujo Tavares, W.: Algoritmos exatos para problema da clique maxima ponderada. Ph.D. thesis, Universidade federal do Ceará (2016). http://www.theses.fr/2016AVIG0211
3. Baz, D.E., Hifi, M., Wu, L., Shi, X.: A parallel ant colony optimization for the maximum-weight clique problem. In: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2016, 23–27 May 2016, Chicago, IL, USA, pp. 796–800. IEEE Computer Society (2016). doi:10.1109/IPDPSW.2016.111
4. Benlic, U., Hao, J.: Breakout local search for maximum clique problems. Comput. OR **40**(1), 192–206 (2013). doi:10.1016/j.cor.2012.06.002
5. Berman, P., Pelc, A.: Distributed probabilistic fault diagnosis for multiprocessor systems. In: Proceedings of the 20th International Symposium on Fault-Tolerant Computing, FTCS 1990, 26–28 June 1990, Newcastle Upon Tyne, UK, pp. 340–346. IEEE Computer Society (1990). doi:10.1109/FTCS.1990.89383
6. Boginski, V., Butenko, S., Shirokikh, O., Trukhanov, S., Gil-Lafuente, J.: A network-based data mining approach to portfolio selection via weighted clique relaxations. Ann. OR **216**(1), 23–34 (2014). doi:10.1007/s10479-013-1395-3
7. Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The maximum clique problem. In: Du, D.Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, pp. 1–74. Springer, Boston (1999). doi:10.1007/978-1-4757-3023-4_1
8. Boussemart, F., Hemery, F., Lecoutre, C., Sais, L.: Boosting systematic search by weighting constraints. In: de Mántaras, R.L., Saitta, L. (eds.) Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI 2004, Including Prestigious Applicants of Intelligent Systems, PAIS 2004, 22–27 August 2004, Valencia, Spain, pp. 146–150. IOS Press (2004)

9. Brockington, M., Culberson, J.C.: Camouflaging independent sets in quasi-random graphs. In: Johnson and Trick [31], pp. 75–88. http://dimacs.rutgers.edu/Volumes/Vol26.html
10. Cai, S., Lin, J.: Fast solving maximum weight clique problem in massive graphs. In: Kambhampati, S. (ed.) Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, 9–15 July 2016, New York, NY, USA, pp. 568–574. IJCAI/AAAI Press (2016). http://www.ijcai.org/Abstract/16/087
11. Carraghan, R., Pardalos, P.M.: An exact algorithm for the maximum clique problem. Oper. Res. Lett. **9**, 375–382 (1990)
12. Chvátal, V.: Resolution search. Discrete Appl. Math. **73**(1), 81–99 (1997). doi:10.1016/S0166-218X(96)00003-0
13. Cohen, D.A., Cooper, M.C., Creed, P., Marx, D., Salamon, A.Z.: The tractability of CSP classes defined by forbidden patterns. J. Artif. Intell. Res. (JAIR) **45**, 47–78 (2012). doi:10.1613/jair.3651
14. Cohen, D.A., Jeavons, P., Jefferson, C., Petrie, K.E., Smith, B.M.: Symmetry definitions for constraint satisfaction problems. Constraints **11**(2–3), 115–137 (2006). doi:10.1007/s10601-006-8059-8
15. Cooper, M.C., Jeavons, P.G., Salamon, A.Z.: Generalizing constraint satisfaction on trees: hybrid tractability and variable elimination. Artif. Intell. **174**(9–10), 570–584 (2010). doi:10.1016/j.artint.2010.03.002
16. Cooper, M.C., Zivny, S.: Hybrid tractable classes of constraint problems. In: Krokhin, A.A., Zivny, S. (eds.) The Constraint Satisfaction Problem: Complexity and Approximability, Dagstuhl Follow-Ups, vol. 7, pp. 113–135. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017). doi:10.4230/DFU.Vol7.15301.4
17. Debroni, J., Eblen, J.D., Langston, M.A., Myrvold, W., Shor, P.W., Weerapurage, D.: A complete resolution of the Keller maximum clique problem. In: Randall, D. (ed.) Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, 23–25 January 2011, San Francisco, California, USA, pp. 129–135. SIAM (2011). doi: 10.1137/1.9781611973082.11
18. Depolli, M., Konc, J., Rozman, K., Trobec, R., Janezic, D.: Exact parallel maximum clique algorithm for general and protein graphs. J. Chem. Inf. Model. **53**(9), 2217–2228 (2013). doi:10.1021/ci4002525
19. Dickerson, J.P., Procaccia, A.D., Sandholm, T.: Optimizing kidney exchange with transplant chains: theory and reality. In: van der Hoek, W., Padgham, L., Conitzer, V., Winikoff, M. (eds.) International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, IFAAMAS, 4–8 June 2012, Valencia, Spain, vol. 3, pp. 711–718 (2012). http://dl.acm.org/citation.cfm?id=2343798
20. Fan, Y., Li, C., Ma, Z., Wen, L., Sattar, A., Su, K.: Local search for maximum vertex weight clique on large sparse graphs with efficient data structures. In: Kang, B.H., Bai, Q. (eds.) AI 2016. LNCS, vol. 9992, pp. 255–267. Springer, Cham (2016). doi:10.1007/978-3-319-50127-7_21
21. Fang, Z., Li, C., Qiao, K., Feng, X., Xu, K.: Solving maximum weight clique using maximum satisfiability reasoning. In: Schaub, T., Friedrich, G., ÓSullivan, B. (eds.) ECAI 2014–21st European Conference on Artificial Intelligence, 18–22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS) 2014. Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 303–308. IOS Press (2014). doi:10.3233/978-1-61499-419-0-303
22. Fang, Z., Li, C., Xu, K.: An exact algorithm based on maxsat reasoning for the maximum weight clique problem. J. Artif. Intell. Res. (JAIR) **55**, 799–833 (2016). doi:10.1613/jair.4953

23. Gendreau, M., Soriano, P., Salvail, L.: Solving the maximum clique problem using a tabu search approach. Ann. OR **41**(4), 385–403 (1993). doi:10.1007/BF02023002
24. Glorie, K., Haase-Kromwijk, B., van de Klundert, J., Wagelmans, A., Weimar, W.: Allocation and matching in kidney exchange programs. Transpl. Int. **27**(4), 333–343 (2014)
25. Gouveia, L., Martins, P.: Solving the maximum edge-weight clique problem in sparse graphs with compact formulations. EURO J. Comput. Optim. **3**(1), 1–30 (2015). doi:10.1007/s13675-014-0028-1
26. Held, S., Cook, W.J., Sewell, E.C.: Maximum-weight stable sets and safe lower bounds for graph coloring. Math. Program. Comput. **4**(4), 363–381 (2012). doi:10.1007/s12532-012-0042-3
27. Hosseinian, S., Fontes, D., Butenko, S.: A quadratic approach to the maximum edge weight clique problem. In: XIII Global Optimization Workshop GOW 2016, pp. 125–128 (2016)
28. Jégou, P.: Decomposition of domains based on the micro-structure of finite constraint-satisfaction problems. In: Fikes, R., Lehnert, W.G. (eds.) Proceedings of the 11th National Conference on Artificial Intelligence, 11–15 July 1993, Washington, DC, USA, pp. 731–736. AAAI Press/The MIT Press (1993). http://www.aaai.org/Library/AAAI/1993/aaai93-109.php
29. Jiang, H., Li, C., Manyà, F.: An exact algorithm for the maximum weight clique problem in large graphs. In: Singh, S.P., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 4–9 February 2017, San Francisco, California, USA, pp. 830–838. AAAI Press (2017). http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14370
30. Johnson, D.S., Trick, M.A.: Introduction to the second DIMACS challenge: cliques, coloring, and satisfiability. In: Cliques, Coloring, and Satisfiability, Proceedings of a DIMACS Workshop, 11–13 October 1993, New Brunswick, New Jersey, USA, [31], pp. 1–10. http://dimacs.rutgers.edu/Volumes/Vol26.html
31. Johnson, D.S., Trick, M.A. (eds.): Cliques, coloring, and satisfiability. In: Proceedings of a DIMACS Workshop, DIMACS/AMS, 11–13 October 1993, New Brunswick, New Jersey, USA. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26 (1996). http://dimacs.rutgers.edu/Volumes/Vol26.html
32. Kumlander, D.: On importance of a special sorting in the maximum-weight clique algorithm based on colour classes. In: An, L.T.H., Bouvry, P., Tao, P.D. (eds.) Modelling, Computation and Optimization in Information Systems and Management Sciences, MCO 2008. Communications in Computer and Information Science, vol. 14, pp. 165–174. Springer, Heidelberg (2008). doi:10.1007/978-3-540-87477-5-18
33. Lau, H.C., Goh, Y.G.: An intelligent brokering system to support multi-agent web-based 4th-party logistics. In: 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 4–6 November 2002, Washington, DC, USA, p. 154. IEEE Computer Society (2002). doi:10.1109/TAI.2002.1180800
34. Malladi, K.T., Mitrovic-Minic, S., Punnen, A.P.: Clustered maximum weight clique problem: algorithms and empirical analysis. Comput. Oper. Res. **85**, 113–128 (2017). http://www.sciencedirect.com/science/article/pii/S0305054817300837
35. Manlove, D.F., O'Malley, G.: Paired and altruistic kidney donation in the UK: algorithms and experimentation. ACM J. Exper. Algorithmics **19**(1) (2014). http://doi.acm.org/10.1145/2670129
36. Mannino, C., Sassano, A.: Solving hard set covering problems. Oper. Res. Lett. **18**(1), 1–5 (1995). doi:10.1016/0167-6377(95)00034-H

37. Mannino, C., Stefanutti, E.: An augmentation algorithm for the maximum weighted stable set problem. Comput. Opt. Appl. **14**(3), 367–381 (1999). doi:10.1023/A:1026456624746
38. Mattei, N., Walsh, T.: Preflib: a library of preference data. In: Perny, P., Pirlot, M., Tsoukiàs, A. (eds.) ADT2013, vol. 8176, pp. 259–270. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41575-3_20. http://www.preflib.org
39. McCreesh, C., Ndiaye, S.N., Prosser, P., Solnon, C.: Clique and constraint models for maximum common (connected) subgraph problems. In: Rueher, M. (ed.) CP 2016. LNCS, vol. 9892, pp. 350–368. Springer, Cham (2016). doi:10.1007/978-3-319-44953-1_23
40. Mehrotra, A., Trick, M.A.: A column generation approach for graph coloring. INFORMS J. Comput. **8**(4), 344–354 (1996). doi:10.1287/ijoc.8.4.344
41. Nogueira, B., Pinheiro, R.G.S., Subramanian, A.: A hybrid iterated local search heuristic for the maximum weight independent set problem. Optim. Lett. 1–17 (2017). doi:10.1007/s11590-017-1128-7
42. Östergård, P.R.J.: A new algorithm for the maximum-weight clique problem. Nord. J. Comput. **8**(4), 424–436 (2001). http://www.cs.helsinki.fi/njc/References/ostergard2001:424.html
43. Östergård, P.R.J.: A fast algorithm for the maximum clique problem. Discrete Appl. Math. **120**(1–3), 197–207 (2002). doi:10.1016/S0166-218X(01)00290-6
44. Prosser, P.: CSPLib problem 063: Winner determination problem (combinatorial auction)
45. Pullan, W.J.: Approximating the maximum vertex/edge weighted clique using local search. J. Heuristics **14**(2), 117–134 (2008). doi:10.1007/s10732-007-9026-2
46. Refalo, P.: Impact-based search strategies for constraint programming. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 557–571. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30201-8_41
47. Roth, A.E., Sönmez, T., Ünver, M.U.: Kidney exchange. Q. J. Econ. **119**(2), 457 (2004). doi:10.1162/0033553041382157
48. Roth, A.E., Sönmez, T., Ünver, M.U.: Efficient kidney exchange: coincidence of wants in markets with compatibility-based preferences. Am. Econ. Rev. **97**(3), 828–851 (2007). http://www.aeaweb.org/articles?id=10.1257/aer.97.3.828
49. Saidman, S.L., Roth, A.E., Sonmez, T., Unver, M.U., Delmonico, F.L.: Increasing the opportunity of live kidney donation by matching for two- and three-way exchanges. Transplantation **81**(5), 773–782 (2006)
50. Sanchis, L.A.: Test case construction for the vertex cover problem. In: Dean, N., Shannon, G.E. (eds.) Computational Support for Discrete Mathematics, Proceedings of a DIMACS Workshop, 12–14 March 1992, Piscataway, New Jersey, USA. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, DIMACS/AMS, vol. 15, pp. 315–326 (1992). http://dimacs.rutgers.edu/Volumes/Vol15.html
51. Sanchis, L.A.: Generating hard and diverse test sets for NP-hard graph problems. Discrete Appl. Math. **58**(1), 35–66 (1995). doi:10.1016/0166-218X(93)E0140-T
52. Sandholm, T., Suri, S.: BOB: improved winner determination in combinatorial auctions and generalizations. Artif. Intell. **145**(1–2), 33–58 (2003). doi:10.1016/S0004-3702(03)00015-8
53. Sandholm, T., Suri, S., Gilpin, A., Levine, D.: CABOB: a fast optimal algorithm for winner determination in combinatorial auctions. Manag. Sci. **51**(3), 374–390 (2005). doi:10.1287/mnsc.1040.0336
54. Sethuraman, S., Butenko, S.: The maximum ratio clique problem. Comput. Manag. Sci. **12**(1), 197–218 (2015). doi:10.1007/s10287-013-0197-z

55. Shimizu, S., Yamaguchi, K., Saitoh, T., Masuda, S.: Fast maximum weight clique extraction algorithm: optimal tables for branch-and-bound. Discrete Appl. Math. **223**, 120–134 (2017). http://www.sciencedirect.com/science/article/pii/S0166218X1730063X
56. Soriano, P., Gendreau, M.: Tabu search algorithms for the maximum clique problem. In: Johnson and Trick [31], pp. 221–244. http://dimacs.rutgers.edu/Volumes/Vol26.html
57. Strash, D.: On the power of simple reductions for the maximum independent set problem. In: Dinh, T.N., Thai, M.T. (eds.) COCOON 2016. LNCS, vol. 9797, pp. 345–356. Springer, Cham (2016). doi:10.1007/978-3-319-42634-1_28
58. Suters, W.H., Abu-Khzam, F.N., Zhang, Y., Symons, C.T., Samatova, N.F., Langston, M.A.: A new approach and faster exact methods for the maximum common subgraph problem. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 717–727. Springer, Heidelberg (2005). doi:10.1007/11533719_73
59. Tavares, W.A., Neto, M.B.C., Rodrigues, C.D., Michelon, P.: Um algoritmo de branch and bound para o problema da clique máxima ponderada. In: Proceedings of XLVII SBPO, vol. 1 (2015)
60. Verfaillie, G., Lemaître, M., Schiex, T.: Russian doll search for solving constraint optimization problems. In: Clancey, W.J., Weld, D.S. (eds.) Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 1996, IAAI 1996, 4–8 August 1996, Portland, Oregon, vol. 1, pp. 181–187. AAAI Press/The MIT Press (1996). http://www.aaai.org/Library/AAAI/1996/aaai96-027.php
61. Wang, Y., Hao, J., Glover, F., Lü, Z., Wu, Q.: Solving the maximum vertex weight clique problem via binary quadratic programming. J. Comb. Optim. **32**(2), 531–549 (2016)
62. Wang, Y., Cai, S., Yin, M.: Two efficient local search algorithms for maximum weight clique problem. In: Schuurmans, D., Wellman, M.P. (eds.) Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 12–17 February 2016, Phoenix, Arizona, USA, pp. 805–811. AAAI Press (2016). http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11915
63. Wang, Y., Cai, S., Yin, M.: Local search for minimum weight dominating set with two-level configuration checking and frequency based scoring function. J. Artif. Intell. Res. (JAIR) **58**, 267–295 (2017). doi:10.1613/jair.5205
64. Wu, Q., Hao, J.: Solving the winner determination problem via a weighted maximum clique heuristic. Expert Syst. Appl. **42**(1), 355–365 (2015). doi:10.1016/j.eswa.2014.07.027
65. Wu, Q., Hao, J., Glover, F.: Multi-neighborhood tabu search for the maximum weight clique problem. Ann. OR **196**(1), 611–634 (2012). doi:10.1007/s10479-012-1124-3
66. Zhou, Y., Hao, J., Goëffon, A.: PUSH: a generalized operator for the maximum vertex weight clique problem. Eur. J. Oper. Res. **257**(1), 41–54 (2017). doi:10.1016/j.ejor.2016.07.056