communications physics

Review article

A Nature Portfolio journal

https://doi.org/10.1038/s42005-026-02606-7

A short review on the maximum clique problem algorithms with classical, AI, and quantum methods

Check for updates

Raffaele Marino [1], Lorenzo Buffoni [1] & Bogdan Zavalnij [2]

This manuscript provides a comprehensive review of the Maximum Clique Problem, a computational problem that involves finding subsets of vertices in a graph that are all pairwise adjacent to each other. As such, this review is a continuation of the series of previous reviews from 1994, 1999 and 2014. The manuscript covers in a simple way classical algorithms and includes a review of recent developments in graph neural networks and quantum algorithms.

Imagine you are at a party with N members, and each participant knows some of the members personally. Your goal is to identify the largest group of friends among the N members, meaning the members who know each other. This is known as the maximum clique problem (MCP).

The MCP is a computational problem that involves finding the largest subset of vertices in a graph that are all pairwise adjacent to each other. In the context of our example, it is to discover the largest group of friends at the party who all know each other. This optimization problem is NP-hard.

The term *clique* originates in social sciences, where complete subgraphs are used to model social cliques (groups of people who all know each other)[1]. The first algorithm to solve this sociological problem can be found in Harary et al.[2].

Clique-finding algorithms have been used in many scientific disciplines including medicine, biology, mathematics, coding theory, economics, financial markets, wireless networks, telecommunications, cryptography, cybersecurity, social science analysis, and physics, among others[3–30].

The clique problem is related to some important combinatorial optimization problems, such as Graph Coloring, maximum independent set (MIS), Minimum Vertex Cover, Graph Clustering, and many others[31–42]. All of these problems can be directly formulated as an MCP or require finding a maximum clique.

Given its fundamental importance in science and its vast number of applications, several influential surveys on the MCP have been published over the years, with the three most notable ones dating back to 1994, 1999, and 2014[43–45].

There has been significant advancement in algorithmic research, particularly in the areas of artificial intelligence (AI) and quantum algorithms. Various experiments and numerical analyses have been conducted on the MCP, which necessitates a comprehensive survey. Therefore, this paper aims to fill this gap by providing a detailed review of recent literature on different approaches to the MCP. Our review will cover classical algorithms and, to the best of our knowledge, include AI and quantum algorithms.

This paper is organized into several sections. First, we present the mathematical formulation of the MCP and the benchmark collections used for testing the algorithms. Second, we describe the classical algorithms for solving the Clique Problem. Third, we provide a general discussion of Graph Neural Networks. Fourth, we review progress by the Quantum Computing community in applying and solving the MCP and related variants. Finally, we conclude our short review with a discussion and outline some open questions for these approaches to this problem.

Maximum clique problem

In this work, if not stated otherwise, we will refer to simple, finite, undirected, unweighted graphs. A graph G is a collection of vertices (also known as nodes or sites) connected by edges (also known as links or arcs). The set of nodes is finite. Formally, a graph is $G(\mathcal{V},\mathcal{E})$. $\mathcal{V}=\{1,2,\ldots,N\}$ represents the set of vertices and $\mathcal{E}$ represents the set of edges. The edges are simply unordered pairs of distinct vertices $\mathcal{E}\subseteq\{\{u,v\},u,v\in V,u\neq v\}$. We will denote $\{i,j\}\in\mathcal{E}$ if there is an edge between nodes i and j, and similarly $\{i,j\}\notin\mathcal{E}$ if there is no such edge. There are no loops or double edges; the edges do not have directions, and there are no weights associated with nodes or edges. Alternatively, a graph G can be described by a $N\times N$ symmetric matrix $A_G=(a_{ij})_{(i,j)\in\mathcal{V}\times\mathcal{V}}$, where $a_{ij}=a_{ji}=1$ if $\{i,j\}\in\mathcal{E}$ is an edge of the graph G, $a_{ij}=a_{ji}=0$ if $\{i,j\}\notin\mathcal{E}$. Such a matrix is called the adjacency matrix of G. The symbol $\overline{G}$ defines the complement graph of G. More precisely, given a graph $G(\mathcal{V},\mathcal{E})$ the complement graph is defined as a graph $\overline{G}(\mathcal{V},\overline{\mathcal{E}})$, where $\overline{\mathcal{E}}=\{\{i,j\}:i,j\in\mathcal{V},i\neq j \text{ and } \{i,j\}\notin\mathcal{E}\}$. In words, a complement graph of a graph $G(\mathcal{V},\mathcal{E})$ is the graph with the same vertex set as G, but with all edges that are not present in G.

A complete graph $G(\mathcal{V},\mathcal{E})$ can also be defined. In such a graph, each pair of distinct vertices is connected by a unique edge. In other words, a

[1]Dipartimento di Fisica e Astronomia, Università degli studi di Firenze, Sesto Fiorentino, Italy. [2]HUN-REN Rényi Institute of Mathematics, Budapest, Hungary. e-mail: bogdan@renyi.hu

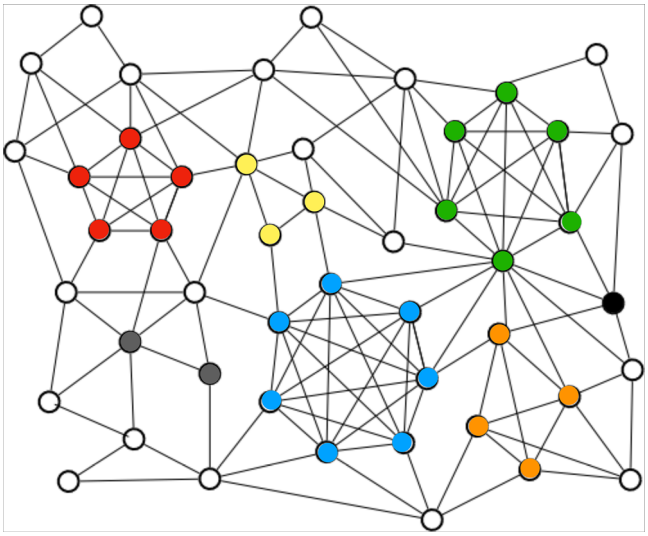


Fig. 1 | An illustrative example of cliques within an undirected graph, showcasing the interconnected nodes and the formation of tightly-knit groups. In black the size of a clique of order 1, in gray of order 2, in yellow of order 3, in orange of order 4, in red of order 5, in green of order 6, and in blue of order 7. This visualization emphasizes the concept of cliques as subsets of vertices within a graph, all of which are adjacent to each other, illustrating the density of connections that define these clusters. The maximum clique in the graph is the set of blue nodes. The blue, green, red, orange, and yellow cliques are also maximal; the black and gray cliques are not.

complete graph is a simple undirected graph where every pair of vertices is adjacent, i.e., $\forall i, j \in \mathcal{V}$ with $i \neq j$ we have $\{i, j\} \in \mathcal{E}$. For a subset $S \subseteq \mathcal{V}$, we call $G(S) = G(S, \mathcal{E} \cap S \times S)$ the subgraph induced by S. We call the set C a clique, if it induces a complete subgraph in $G(\mathcal{V}, \mathcal{E})$, that is $C \subseteq \mathcal{V}$ such that $G(C)$ is complete (see Fig. 1). We call the size of the clique the cardinality of C, i.e., $|C|$.

Given a node $v \in \mathcal{V}$, $N(v)$ denotes the neighborhood (or frontier) of v, i.e., the set of all nodes adjacent to v: $N(v) = \{j \in \mathcal{V} : \{v, j\} \in \mathcal{E}\}$.

We can define coloring of nodes by k colors, that is, giving each node numbers from $1, \ldots, k$ as a label. Formally, the coloring of the nodes of G can be described by a map f: $V \to \{1, \ldots, k\}$. We call a coloring proper or legal if two adjacent nodes receive different colors. That is if $i, j \in \mathcal{V}, i \neq j, \{i, j\} \in \mathcal{E}$ then $f(i) \neq f(j)$.

Closely related to cliques are the notions of independent sets and vertex covers in graphs. An independent set, or in other words, a stable set, is a set of nodes that has no edge among them. A vertex cover is a set of nodes such that every edge in the graph has at least one endpoint among these nodes. If given a graph $G(\mathcal{V}, \mathcal{E})$ and an independent set I, the set $K = \mathcal{V} \backslash I$ is a covering set—presume that there would be an edge that has no endpoints in K, that would mean the endpoints of this edge are in I, which contradicts I being an independent set. If given a graph $G(\mathcal{V}, \mathcal{E})$ and a covering set K, the set $I = \mathcal{V} \backslash K$ is an independent set—as for every edge of the graph G, at least one endpoint lies in K, so there cannot be any edges between any two nodes of I.

We can now define some problems related to cliques.

Problem 1. Given a graph $G(\mathcal{V}, \mathcal{E})$ and an integer k, decide if graph G contains a clique of size k.

This problem is called the k-clique problem. Sometimes it is also required to present a k-clique as well.

Problem 2. Given a graph $G(\mathcal{V}, \mathcal{E})$ find the maximum value of k, so that the graph G has a clique of size k.

This problem is called the MCP. The size of a maximum clique is also called the clique number of the graph, and denoted in graph theory as $\omega(G)$. In other words, formally, $\omega(G) = \max\{|C| : C \text{ is a clique in } G\}$. Clearly, solving problem 2 solves problem 1 as well. Also, a small sequence of k-clique search would solve the MCP. In addition, one may also require that, apart from calculating the clique number, the program should present one of the maximum cliques as well. This is a slightly different problem, as in theory it would be possible to solve the first without actually presenting the clique in question. Although, to our knowledge, all programs actually also find and present a maximum clique.

Compared to the maximum clique, one can speak about *maximal* cliques. A maximal clique is a clique that cannot be extended; that is, no node can be added, so the resulting set of nodes is still a clique. A maximum clique is by definition always a maximal one, but not vice versa.

Problem 3. Given a graph $G(\mathcal{V}, \mathcal{E})$ list all maximum cliques in the graph G.

This problem is clearly the most demanding one, and solving it, we got a solution to all previous ones.

There are equivalent problems to MCP. Finding a maximum clique in the complement of a graph $G(\mathcal{V}, \mathcal{E})$ is equivalent to finding a maximum set of vertices in $\overline{G}(\mathcal{V}, \overline{\mathcal{E}})$ whose elements are pairwise nonadjacent, thus a MIS denoted by $\alpha(G)$. Given a MIS I in a graph $G(\mathcal{V}, \mathcal{E})$, the set of sites obtained by the relation $\mathcal{V} \backslash I$ is a Minimum Vertex Cover set (MVC). Given that, the MIS, MVC, and MCP are equivalent problems from a mathematical perspective. In the following sections, we will also discuss MIS and MVC results if we believe they are useful.

The MCP is well studied from the complexity viewpoint, with early complexity results on the problem being reviewed in Pardalos and Xue[43], Bomze et al.[44], and Wu and Hao[45]. However, during the last decade, a number of advances in the research of the computational complexity of the MCP appeared[46]. Although a full review of these theoretical results is clearly beyond the scope of this paper, we provide a brief summary of some main results.

Hardness of the problem

One can raise the question about how hard the proposed problem is from the point of view of practical computer science. First, we will examine the hardness of the exact solution of the problem. Second, we would like to discuss how effective a heuristic approach can be for this problem. Both approaches, the exact and the heuristic, are telling us that this problem is notoriously hard.

Evaluating the *exact approach*, we would like to point to the historical fact that the decision formulation of MCP, the k-clique problem, problem 1, as stated in the introduction, was first shown to be NP-complete in the seminal work of Karp[47]. Thus, its optimization variant, i.e., MCP, problem 2, where the goal is to find a clique of the largest possible size, is NP-hard. Given this result, the usual answer to the hardness of these problems is that they are exponential. But there are several factors here.

First, that this problem class, namely the worst-case running time of these problems, would be exponential is only a hypothesis, named the Exponential Time Hypothesis (ETH)[48]. So even in the case of P ≠ NP, it may turn out that the complexity of these problems is still sub-exponential.

Second, obviously, different kinds of problems or even problem instances can be very easy or very hard. As even the ETH is only stating exponential running time for the worst case. So, consequently, for algorithm engineering, it is of utmost importance to have a good set of benchmarks for testing. A usual metric of the graph that can sometimes, but not always, point to the hardness is the density of the graph. As shown in the outstanding paper of Scottish researchers, densities around 80–95% are usually the hardest, and this research is also backed up with the experimental results of different clique search algorithms[49].

Third, the problems stated above are inherently different. Take, for example, the k-clique problem, problem 1. On one hand, if a heuristic algorithm finds a clique of size k one has already solved the problem. Also, if someone is using a heuristic way to find an upper bound, and that bound is less than k, this also solves the problem—there cannot be a k-clique in the graph. On the other hand, if there is no k-clique present in the graph, and one cannot find an upper bound to prove it, then solving this problem will be hard.

For the MCP, finding a big clique by heuristics does not solve the problem alone, only if there is also an upper bound such that the value of the bound is equal to the size of the found clique. In all other cases, one needs to find the maximum clique *and* prove that there is no bigger clique present—and this is hard.

As for problem 3, listing all maximum cliques, there seems to be no shortcut, no heuristics to help solution, the problem is always hard.

Finally, there is a curious and sometimes confusing difference between MVC and MCP, although we stated that they are the same from a mathematical point of view. MVC is famous for being fixed-parameter tractable (FPT), while MCP is *not*, see refs. 50,51. (Formally, the k-clique problem is in the W[1] complexity class. The reader can find the more technical definition on the W hierarchy and the fixed parameter tractability in refs. 50,51). Recall that FPT means that the problem can be solved in time $f(k) \cdot |x|^{O(1)}$ for some computable function f. In our case, x is the size of the graph, that is $|\mathcal{V}|$, and k would be the size of a minimum vertex cover set, or the size of a maximum clique. Though if K is a covering set, the set $I = \mathcal{V} \backslash K$ is an independent set, and $C = \mathcal{V} \backslash K$ is a clique in the complement graph $\overline{G}$. That is $|K| = |\mathcal{V}| - |C|$. Which means that if MVC is small, that is $k = |K|$, $k \ll |V|$, the parameterized problem is dominated by the $f(k)$ function. The corresponding clique problem, though, has a solution where almost all nodes are in the clique. In simple words, MVC is FPT for small set sizes, while the clique problem is FPT for clique number close to the size of the set of all nodes—with the parameter value $(|\mathcal{V}| - k)$. For small clique numbers, the problem is clearly not FPT; that is why the clique problem is not in the FPT class (by the parameter of the clique size).

Given that the exact solution of the MCP is computationally hard, one would decide to use a *heuristic approach* in order to find a suitably good solution instead of a proven one. That is to find a big enough (maximal) clique, and hope that the size of this clique would be close to the size of a maximum clique. This approach is driven by the fact that some real-world problems require finding a big enough clique and do not need a proof of optimality. But this approach is also hard by itself, as it is proven that even approximating the clique number is hard[52]. This means that, opposed to other problems, such as the Knapsack Problem or the metric Traveling Salesman Problem, where there are effective constant-factor approximation algorithms, the approximation of MCP cannot be as effective.

The state-of-the-art of the best-known polynomial-time approximation algorithm is due to Feige[53], which yields an approximation ratio of $O(N(\log\log N)^2/(\log N)^3)$. The approximation ratio is the ratio between the size of the maximum clique and the size of the solution output by the algorithm. In Engebretsen et al.[54], however, the authors showed that the MCP is not approximable within a factor of $N/2^{O(\log N)/\sqrt{\log\log N}}$ under the assumption that NP $\subseteq$ ZPTIME$(2^{O(\log N)(\log\log N)^{3/2}})$. An improved result shows that the MCP is not approximable within $N^{1-\epsilon}$, with $\epsilon > 0$, unless P=NP[55].

In Chitnis et al.[56], it is established that unless NP $\subseteq$ SUBEXP, for any $0 < \delta < 1$ there exists a constant that depends by δ, i.e., $F(\delta) > 0$, such that the MCP has no Fixed-Parameter Tractable (FPT) optimum approximation with ratio $\rho(OPT) = OPT^{1-\delta}$ in $2^{OPT^F} poly(|\mathcal{V}|)$. Finally, in 2020, the authors in Chalermsook et al.[57] showed that there is no nontrivial FPT approximation algorithm for MCP. More precisely, there is no $o(OPT)$-FPT-approximation algorithm for MCP (for an updated summary on MIS, we refer to Marino et al.[37]).

Research in this direction came up with an interesting approach, which in the last decade has obtained attention from the scientific community and has been barely mentioned in the reviews[44,45]: the hidden clique problem (HCP), or planted clique model. It has received a great deal of attention and has become a hallmark of a research area that is now called the study of statistical-computational gaps. These are the regimes where a signal is detectable or recoverable yet believed to be intractable for polynomial-time algorithms. A common strategy to provide evidence for such a gap is to prove that powerful classes of efficient algorithms are unable to solve the planted problem in the (conjecturally) hard regime.

Several authors have proposed that the search for powerful, effective clique-finding algorithms could be expressed as a challenge, perhaps to attract the widest set of challengers to the problem. Mark Jerrum, in his 1992 paper[58] sets out several of these. His paper shows that a restricted version of stochastic search is unlikely to reach a maximum clique, and also introduces the additional problem of finding an artificially hidden clique. A hidden clique or *planted solution* is just what it sounds like, a single subgraph of sites, with $|C_{HC}| > |C_{\max}|$, so that it can be distinguished, for which all the missing bonds among those sites have been restored. A series of papers[59,60] show the existence of polynomial algorithms that are able to compute a hidden clique of size $|C_{HC}|$ proportional to $N^{0.5}$. The state-of-the-art for the HCP is given by Deshpande et al.[61]. They developed a rigorous analysis that is asymptotically exact as $N \to \infty$, and they prove that their algorithm is able to find hidden cliques of size $|C_{HC}| \geq \sqrt{N/e}$ with high probability.

For computing lower bounds, a very useful method has been established in recent years. The name of such a method is the sum of squares (SOS) semidefinite hierarchy. SOS algorithms are particularly attractive targets for lower bounds because of their broad applicability and strong guarantees[62].

The sum-of-squares (SOS) hierarchy, also known as the Lasserre hierarchy[63], is a hierarchy of convex relaxations of increasing power and increasing computational cost. For each integer $r \geq 1$, the corresponding convex relaxation is known as the r-th round of the SOS hierarchy. The first round ($r = 1$) coincides with basic semidefinite programming, or to sum-of-squares optimization over polynomials of degree at most 2. Passing from level r to $r + 1$ adds the program with variables and constraints so that certificates may involve polynomials up to degree $2(r + 1)$. In ref. 64, the authors proved the first average-case lower bound for HCP. They showed that for $r = 1$, SOS can find a solution up to $|C_{HC}| = \Theta(\sqrt{N})$. Deshpande et al.[65], instead, considered the $r = 4$ SOS relaxation. This is the first level at which the SOS hierarchy differs substantially from the baseline spectral algorithm of Alon, Krivelevich, and Sudakov[59]. The authors in ref. 65 proved that SOS fails unless $|C_{HC}| \geq cN^{1/3}/\log N$. Their proof uses the moment method to bound the spectrum of a certain random association scheme, i.e., a symmetric random matrix whose rows and columns are indexed by the edges of an Erdős-Rényi random graph.

A full review of these theoretical results is clearly beyond the scope of this manuscript; however, one can refer to refs. 66–69, and references therein for a more accurate description of the SOS methodology.

Mathematical programming formulation of MCP

One obvious way to solve the MCP is to use a mathematical programming solver, such as an Integer Linear Programming (ILP) solver.

The formulation of the MCP has been the subject of many studies, which offer valuable insights into the problem and its theoretical and practical implications. Reviewing these studies is important for gaining a comprehensive understanding of existing MCP formulations and discovering new results. Bomze et al.[44], and Pardalos and Xue[43] provide an in-depth analysis of existing formulations, while we will focus on some common and recently developed formulations.

Following refs. 44,45,70, the MCP can be formulated in the following manner: let's $x_i \in \{0, 1\}$, $i = 1, \ldots, N$, we consider to find the max $\sum_{i=1}^{N} x_i$ under the constraint that $x_i + x_j \leq 1, \forall \{i, j\} \in \overline{\mathcal{E}}$, which is usually called the edge formulation.

In this simple formulation, any feasible solution of MCP on a graph G is obtained by the set of vertices C such that a vertex $i \in C$ if $x_i = 1$ and $x_i = 0$ otherwise. Nemhauser and Trotter[71,72] showed that if a variable $x_i = 1$ holds for an optimal solution to the linear relaxation of the above formulation, then $x_i = 1$ holds for at least one optimal solution to the integer formulation. Clearly, this result can be used by an algorithm to reduce the search space explored when seeking an optimal clique.

The same result can be obtained using an alternative formulation based on independent sets. By denoting with S the set of all maximal independent sets in G, any clique C in G can contain no more than a single vertex from such a set. Mathematically, one wants to maximize $\sum_{i=1}^{N} x_i$, with

$x_i \in \{0, 1\}$, $i = 1, \ldots, N$, under the constraint $\sum_{i \in s} x_i \leq 1$, $\forall\, s \in S$. This formulation offers the benefit of a minimal discrepancy between an optimal solution and its linear relaxation. However, cataloging all independent sets within a generic graph presents a challenge, as the number of independent sets increases exponentially, corresponding to the size of the graph[73]. Thus, applying the relaxed version of the independent set formulation is a non-trivial task, and probably calls for a column generation approach, which is out of our scope.

The reformulation of the problem of maximum weighted clique is straightforward using any previously detailed methods. The only difference is that one needs to add weights to the objective function. The edge-weighted version of the problem is far more challenging. For sparse graphs, using variables representing edges can be a viable solution[74]. For dense graphs, other approaches may be needed, such as using quadratic programming[75]. The quadratic reformulation as an alternative technique can be used for the node weighted version as well, and it can, in some cases, generate good computational results[76]. For connections between the MCP and a certain standard quadratic programming problem, we refer to ref. 77.

Remarkable discretized formulations of MCP were presented in Martins[78]. By using additional and extra indexed variables and constraints, the formulations depend on the range of variation of an interval containing the clique number of the graph. Therefore, tight lower and upper bounds may strongly benefit the dimension of the formulations. Lower bounds identify cliques that an efficient algorithm can find, while upper bounds are on the actual maximum clique, not just on the size an algorithm can find. A simple application of the probabilistic method can reveal with a high degree of certainty that the maximum clique, in an Erdős-Rényi graph $G(|\mathcal{V}| = N, p)$ with probability p that two vertices are linked to each other, approximates to $d(N) = 2\log_{1/p} N - 2\log_{1/p}\log_{1/p} N + 2\log_{1/p}\frac{e}{2} + 1 + O(1)$ for large N[79]. Matula, instead, drew attention to what is now termed a concentration result for the MCP. As $N \to \infty$ the sizes of the largest cliques $C_{\max}$ that will occur are concentrated on just two values, the integers immediately below and above $|C_{\max}| = d(N)$. To do this, he used the second moment of the distribution of the numbers of cliques of a certain size to bound the fraction of graphs with no such cliques and sharpened the result of Matula[80] by computing a weighted second moment. Indeed, Markov's inequality provides upper bounds, and Chebyshev's inequality provides lower bounds on the existence of such cliques[80–82]. The fraction of graphs $G(N, p)$ with maximum clique size $|C_{\max}|$, is bounded as follows:

$$\left(\sum_{j=\max\{0,2k-N\}}^{k} \frac{\binom{N-k}{k-j}\binom{k}{j}}{\binom{N}{k}} p^{-\binom{j}{2}}\right)^{-1} \leq P(|C_{\max}| \geq k) \leq \binom{N}{k} p^{\binom{k}{2}}. \tag{1}$$

Recently, in Belachew et al.[83], it was formulated a very convenient continuous characterization of the MCP based on the symmetric rank-one non-negative approximation of a given matrix and building a one-to-one correspondence between stationary points and cliques of a given graph G. In particular, the authors showed that the local (resp. global) minima of the continuous problem correspond to the maximal (resp. maximum) cliques of the given graph G.

The last formulation of the MCP is the one best suited for quantum computers in the form of a Quadratic Unconstrained Binary Optimization (QUBO) problem. To do that, we first take the trivial mapping from MCP to MIS on the dual graph. Then, given a graph $G = (V, E)$, in order to find the MIS, we need to assign a binary variable x_i to each one of the i vertices, which is 1 if the vertex belongs to the MIS and 0 otherwise. Thus, this problem becomes:

$$\max_{x_i \in \{0,1\}} \sum_{i=1}^{N} x_i \tag{2}$$

subjected to the constraint

$$\sum_{(i,j) \in E} x_i x_j = 0, \tag{3}$$

The equivalent QUBO representation of this problem, expressed in Hamiltonian form, is

$$H = -A \sum_{i=1}^{N} x_i + B \sum_{(i,j) \in E} x_i x_j, \tag{4}$$

where the coefficients (penalties) A and B can be chosen appropriately as discussed in refs. 84,85.

Benchmark collections

As we already pointed out, algorithm design is always going along with the question of testing. Historically, algorithms for MCP were tested on random graphs based on the Erdős-Rényi model. Such a graph, $G(N, p)$, is a graph of N nodes. For each pair of nodes, there is an edge with probability p included independently from every other edge. This graph class is easy to generate and so test algorithms against it. But these graphs are far from perfect for testing, as they lack any structure.

In evaluating the performance of algorithms designed for the MCP, many benchmarks appeared in the academic literature. In the following, we list the most commonly used benchmarks for testing algorithms.

DIMACS. The DIMACS benchmark from 1996 was part of the second DIMACS Implementation Challenge[86]. This conference focused on problems related to Clique, Satisfiability, and Graph Coloring. This challenge led to the creation of a set of standard benchmark graphs that have been extensively used to evaluate algorithms for these problems. It is composed of 80 graphs, of different order and size, as presented in Jin et al.[87]. Note that there are artificially constructed graphs that thus have a known clique number, but still cannot be solved by a recent exact solver. Such are, for example, the graphs coming from coding theory, like some of the Hamming graphs and Johnson graphs[88,89].

The DIMACS benchmark encompasses a diverse array of practical challenges, including those derived from coding theory, fault diagnosis, and Keller's conjecture, among others. This collection not only features graphs generated through random processes but also includes graphs specifically designed to obscure the maximum clique by integrating vertices of lower degrees. The scale of these DIMACS instances is quite broad, with some comprising fewer than 50 vertices and 10^3 edges, while others expand to over 3300 vertices and 5 million edges. Based on documented findings, the majority of the 80 instances in the DIMACS benchmark have had their maximum cliques identified. However, some exceptions persist. Today, experimental results obtained by an exact algorithm show that the number of instances solved to proven optimality is 77[90]. A list of best-known solutions in literature for the DIMACS benchmark is given in Table 1. In this table, and also in the following ones, we indicated the name of the file, the number of nodes ($|\mathcal{V}|$), the number of edges ($|\mathcal{E}|$), and the size of the maximum clique ($\omega(G)$).

BHOSLIB. The BHOSLIB benchmark is a collection of graph instances specifically designed for testing several clique algorithms. The BHOSLIB benchmark is composed of 41 graphs with hidden optimal solutions[91]. The instances in BHOSLIB are known for being particularly challenging. They are translated from hard random SAT instances generated at the exact phase transition of the RB model[92]. The size of these instances

Table 1 | DIMACS best known results in literature for $\omega(G)$ (optimal values are marked with '*')

Name	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\omega(G)$
brock200_1	200	14834	21*
brock200_2	200	9876	12*
brock200_3	200	12048	15*
brock200_4	200	13089	17*
brock400_1	400	59723	27*
brock400_2	400	59786	29*
brock400_3	400	59681	31*
brock400_4	400	59765	33*
brock800_1	800	207505	23*
brock800_2	800	208166	24*
brock800_3	800	207333	25*
brock800_4	800	207643	26*
C125.9	125	6963	34*
C250.9	250	27984	44*
C500.9	500	112332	≥57
C1000.9	1000	450079	≥68
C2000.5	2000	999836	16*
C2000.9	2000	1799532	≥80
C4000.5	4000	4000268	18*
DSJC500.5	500	125248	13*
DSJC1000.5	1000	499652	15*
keller4	171	9435	11*
keller5	776	225990	27*
keller6	3361	4619898	59*
MANN_a9	45	918	16*
MANN_a27	378	70551	126*
MANN_a45	1035	533115	345*
MANN_a81	3321	5506380	1100*
hamming6-2	64	1824	32*
hamming6-4	64	704	4*
hamming8-2	256	31616	128*
hamming8-4	256	20864	16*
hamming10-2	1024	518656	512*
hamming10-4	1024	434176	40*
gen200-p0.9-44	200	17910	44*
gen200-p0.9-55	200	17910	55*
gen400-p0.9-55	400	71820	55*
gen400-p0.9-65	400	71820	65*
gen400-p0.9-75	400	71820	75*
c-fat200-1	200	1534	12*
c-fat200-2	200	3235	24*
c-fat200-5	200	8473	58*
c-fat500-1	500	4459	14*
c-fat500-2	500	9139	26*
c-fat500-5	500	23191	64*
c-fat500-10	500	46627	126*
johnson8-2-4	28	210	4*
johnson8-4-4	70	1855	14*
johnson16-2-4	120	5460	8*
johnson32-2-4	496	107880	16*

Table 1 (continued) | DIMACS best known results in literature for $\omega(G)$ (optimal values are marked with '*')

Name	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\omega(G)$
p_hat300-1	300	10933	8*
p_hat300-2	300	21928	25*
p_hat300-3	300	33390	36*
p_hat500-1	500	31569	9*
p_hat500-2	500	62946	36*
p_hat500-3	500	93800	50*
p_hat700-1	700	60999	11*
p_hat700-2	700	121728	44*
p_hat700-3	700	183010	62*
p_hat1000-1	1000	122253	10*
p_hat1000-2	1000	244799	46*
p_hat1000-3	1000	371746	68*
p_hat1500-1	1500	284923	12*
p_hat1500-2	1500	568960	65*
p_hat1500-3	1500	847244	94*
san200_0.7_1	200	13930	30*
san200_0.7_2	200	13930	18*
san200_0.9_1	200	17910	70*
san200_0.9_2	200	17910	60*
san200_0.9_3	200	17910	44*
san400_0.5_1	400	39900	13*
san400_0.7_1	400	55860	40*
san400_0.7_2	400	55860	30*
san400_0.7_3	400	55860	22*
san400_0.9_1	400	71820	100*
san1000	1000	250500	15*
sanr200_0.7	200	13868	18*
sanr200_0.9	200	17863	42*
sanr400_0.5	400	39984	13*
sanr400_0.7	400	55869	21*

varies, starting from graphs with 450 vertices and 83198 edges, and scaling up to graphs with 4000 vertices and 7 million edges. Although designed for a heuristic approach, these graphs were used as test graphs for different exact algorithms. It was found that only a limited number of such algorithms were able to effectively address them. On the other hand, several recent heuristic algorithms can attain the known optimal solutions for these instances with no particular difficulty[93–97]. For comprehensive information on this benchmark and to access the downloadable instances, please visit the following website: BHOSLIB Benchmark. For the sake of simplicity, here we report the list of the hidden solution size in Table 2.

EVIL. The Evil benchmark[98] is notable for its ability to create challenges of varying difficulty levels for maximum clique search algorithms. This adaptability is crucial because many existing benchmark problems are either too simple for modern solvers or excessively challenging. The Evil benchmark addresses this gap by offering a range of tests that can be fine-tuned for different degrees of difficulty. The benchmark's design intentionally widens the gap between the chromatic number and the clique number of the graphs, while using the hidden clique technique in a (partially) random graph, making it a valuable tool for algorithm developers and researchers. This aspect of the benchmark allows for a

Table 2 | BHOSLIB benchmark with size of planted clique of size $\omega(G)$ (optimal values are marked with '*')

Name	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\omega(G)$
frb30-15-1	400	83198	30*
frb30-15-2	400	83151	30*
frb30-15-3	400	83216	30*
frb30-15-4	400	83194	30*
frb30-15-5	400	83231	30*
frb35-17-1	595	148859	35*
frb35-17-2	595	148868	35*
frb35-17-3	595	148784	35*
frb35-17-4	595	148873	35*
frb35-17-5	595	148572	35*
frb40-19-1	760	247106	40*
frb40-19-2	760	247157	40*
frb40-19-3	760	247325	40*
frb40-19-4	760	246815	40*
frb40-19-5	760	246801	40*
frb45-21-1	945	386854	45*
frb45-21-2	945	387416	45*
frb45-21-3	945	387795	45*
frb45-21-4	945	387491	45*
frb45-21-5	945	387461	45*
frb50-23-1	1150	580603	50*
frb50-23-2	1150	579824	50*
frb50-23-3	1150	579607	50*
frb50-23-4	1150	580417	50*
frb50-23-5	1150	580640	50*
frb53-24-1	1272	714129	53*
frb53-24-2	1272	714067	53*
frb53-24-3	1272	714229	53*
frb53-24-4	1272	714048	53*
frb53-24-5	1272	714130	53*
frb56-25-1	1400	869624	56*
frb56-25-2	1400	869899	56*
frb56-25-3	1400	869921	56*
frb56-25-4	1400	869262	56*
frb56-25-5	1400	869699	56*
frb59-26-1	1534	1049256	59*
frb59-26-2	1534	1049648	59*
frb59-26-3	1534	1049729	59*
frb59-26-4	1534	1048800	59*
frb59-26-5	1534	1049829	59*

more comprehensive assessment of the algorithms' capabilities, especially in scenarios where traditional benchmarks might not provide a sufficient challenge or might not accurately represent the complexity of real-world problems. For the presented work, we extended the original set of graphs to conclude some more challenging instances for the novel algorithms. The size of the new set of instances varies, starting from graphs with 120 vertices and 6595 edges, and scaling up to graphs with 750 vertices and 270928 edges. In summary, the Evil benchmark for the MCP offers a versatile and challenging set of small test instances, allowing for a thorough and nuanced evaluation of maximum clique search algorithms. This benchmark stands out for its ability to provide a wide range of difficulties, making it a valuable resource in the field of algorithmic research and development. The benchmark is composed of 40 graphs. For the sake of simplicity, here we report them in Table 3 To download the instances, please visit: Evil Benchmark graphs

Benchmarks from coding theory. Other instances are also widely used in the MCP algorithm development community, such as instances arising from coding theory[99,100]. Here, the question is to find a maximum size code under some channel restrictions, such as Single-Deletion-Correcting Codes, Two-Deletion-Correcting Codes, Codes For Correcting a Single Transposition, etc. The task is to find the MIS, so in the present work, we list the properties of the complement graphs for the MCP. We list these instances in Table 4, some instances have optimal value proven, but for some instances, we have only lower and upper bounds. These instances greatly vary in size and hardness; some are still unsolved. They are considered very challenging problems.

A miscellaneous benchmark list. In 2019, the 4th Parameterized Algorithms and Computational Experiments (PACE) challenge[101] notably featured a track on the Minimum Vertex Cover (MVC) problem. The challenge, organized by the PACE community, provided a significant contribution to the computational research community in the form of benchmarks. These benchmarks are particularly valuable. They offer a comprehensive set of instances that can be repurposed for testing and evaluating algorithms for the MCP. Specifically, the PACE 2019 challenge's Vertex Cover instances encompass a diverse range of scenarios, catering to the assessment of various algorithmic strategies. This diversity stems from the inclusion of instances originating from different publicly available graph sets and reductions from other combinatorial hard problems. The benchmarks, which are publicly accessible through the PACE Challenge website Pace Benchmark, consist of 200 instances, each offering unique challenges and insights for the MCP. The set was collected from various sources: SAT benchmarks, BHOSLIB benchmarks, DIMACS benchmarks, and other sources[102].

Another intriguing benchmark that can be adapted for the MCP is the SUDOKU Benchmark. This benchmark comprises an extensive set of ~50,000 Sudoku puzzles. A Sudoku puzzle features a 9×9 grid, divided into nine 3×3 subgrids, with some cells pre-filled. The goal is to fill the remaining cells so that each row, column, and subgrid contains each digit from 1 to 9 exactly once. In the context of the MCP, the vertices represent specific numbers and squares of the Sudoku grid. An edge is formed between two vertices if the corresponding (number, square) pairs adhere to Sudoku's rules. Each Sudoku instance in this benchmark has a solution, offering a rich dataset for MCP research[103–105]. Massive and sparse datasets are also used for testing algorithms, like GNN. Examples of such datasets are IMDB[106], COLLAB[107], TWITTER[108], and FACEBOOK[109].

These benchmarks, with their inherent complexity and diversity, serve as excellent tools for researchers aiming to develop and refine algorithms for the MCP. Their varied nature allows for a comprehensive evaluation of algorithmic performance across different instance characteristics, thereby contributing significantly to the advancement of research in this domain.

Quantum benchmark. In the Quantum Computing domain, the problem of benchmarks is one that goes beyond the specific case of MCP or MIS. Indeed, the variety of experimental platforms, constraints, and ways of performing computation makes standardized benchmarking a painful (albeit often necessary) task. Regarding the specific problems we are analyzing in this review, we have seen how each platform works best for problems designed specifically to fit well its own connectivity, but there are no reference benchmarks, albeit some theoretical results do exist[110–112].

In light of the diverse challenges associated with benchmarking in the quantum computing field–stemming from the variety of experimental platforms, operational constraints, and computational methodologies–we

Table 3 | Evil benchmarks with size of planted clique of size $\omega(G)$ (optimal values are marked with '*')

Name	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\omega(G)$
chv12 × 10	120	6595	20*
myc5 × 24	120	6904	48*
myc11 × 11	121	6752	22*
s3m25 × 5	125	6877	20*
myc23 × 6	138	8211	12*
myc5 × 30	150	10837	60*
s3m25 × 6	150	10073	24*
myc11 × 14	154	11080	28*
chv12 × 15	180	15166	30*
myc5 × 36	180	15671	72*
myc23 × 8	184	15072	16*
myc11 × 17	187	16490	34*
s3m25 × 8	200	18350	32*
myc5 × 42	210	21404	84*
myc11 × 20	220	22960	40*
myc23 × 10	230	24072	20*
chv12 × 20	240	27328	40*
myc5 × 48	240	27962	96*
s3m25 × 10	250	29075	40*
myc11 × 23	253	30422	46*
chv12 × 30	360	62211	60*
chv12 × 40	480	111059	80*
chv12 × 60	720	251269	120*
myc5 × 60	300	43659	120*
myc5 × 80	400	77893	160*
myc5 × 100	500	121781	200*
myc5 × 120	600	175458	240*
myc5 × 140	700	239044	280*
myc11 × 30	330	52214	60*
myc11 × 40	440	93327	80*
myc11 × 50	550	146280	100*
myc11 × 60	660	210997	120*
myc23 × 15	345	55464	30*
myc23 × 20	460	99896	40*
myc23 × 25	575	157359	50*
myc23 × 30	690	227595	60*
s3m25 × 15	375	66480	60*
s3m25 × 20	500	119354	80*
s3m25 × 25	625	187499	100*
s3m25 × 30	750	270928	120*

Table 4 | Graphs arising from coding theory (optimal values are marked with '*')

Name	$\lvert\mathcal{V}\rvert$	$\lvert\bar{\mathcal{E}}\rvert$	$\omega(G)$
1dc.64	64	1473	10*
1dc.128	128	6657	16*
1dc.256	256	28801	30*
1dc.512	512	121089	52*
1dc.1024	1024	499713	94*
1dc.2048	2048	2037761	172*
2dc.128	128	2955	5*
2dc.256	256	15457	7*
2dc.512	512	75921	11*
2dc.1024	1024	354614	16*
2dc.2048	2048	1591677	24*
1tc.8	8	22	4*
1tc.16	16	98	8*
1tc.32	32	428	12*
1tc.64	64	1824	20*
1tc.128	128	7616	38*
1tc.256	256	31328	63*
1tc.512	512	127552	110*
1tc.1024	1024	515840	196*
1tc.2048	2048	2077184	352*
1et.64	64	1752	18*
1et.128	128	7456	28*
1et.256	256	30976	50*
1et.512	512	126784	100*
1et.1024	1024	514176	171*
1et.2048	2048	2073600	316*
1zc.128	128	5888	18*
1zc.256	256	27008	36*
1zc.512	512	116992	62*
1zc.1024	1024	490496	112–117
1zc.2048	2048	2017280	198–210
1zc.4096	4096	8202240	379–410

introduce a novel benchmark tailored to accommodate the capabilities of contemporary quantum computing hardware. This new benchmark is designed to bridge the gap in standardized benchmarking by providing a set of instances specifically crafted to challenge today's quantum computers. We release these instances, confident in their capacity to serve as robust benchmarks. The difficulty of these instances is primarily attributed to the minimal energy differences between the ground states (the planted solutions) and the first excited states. This subtle energy gap ensures that identifying the ground state becomes a sufficiently challenging task for quantum computing systems, thus testing their limits and highlighting their capabilities.

The intricacies of this hardness lie in the quantum computational principle of finding the lowest energy state, or the ground state, which corresponds to an optimal solution for the quantum algorithms. In classical computing, similar challenges are encountered in optimization problems where finding the global minimum in a landscape of local minima can be arduous. In the quantum context, the challenge is magnified by the great susceptibility of quantum systems to all sources of noise, making it difficult to distinguish between closely spaced energy levels and maintain the system in a highly coherent low-energy state. The instances we introduce are thus specifically designed to exploit these properties.

We extended these instances with small instances from the coding theory benchmark, the EVIL benchmark, and the DIMACS set. The set of instances, all of them of order at most 128, and their corresponding details, including the sizes of the solutions, are made available at Quantum Benchmark graphs, and a comprehensive list with the relative solution sizes is presented in Table 5. By offering these instances to the quantum computing community, we aim to establish a benchmark that not only reflects the current state of quantum hardware but also pushes the envelope in quantum computing capabilities, driving the development of more sophisticated and powerful quantum systems. Importantly, as the size of

Table 5 | Benchmarks for quantum algorithms (optimal values are marked with '*')

Name	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\omega(G)$
g136 × 6	60	1716	48*
g136 × 8	80	3087	64*
g136 × 10	100	4827	60*
g136 × 12	120	6970	96*
g150 × 6	60	1735	54*
g150 × 8	80	3097	72*
g32559 × 6	66	2075	48*
g32559 × 8	88	3702	64*
g32559 × 10	110	5837	80*
g32565 × 6	66	2048	42*
g32565 × 8	88	3670	56*
g32565 × 10	110	5758	70*
g33896 × 4	72	2228	28*
g33896 × 5	90	3575	35*
g33896 × 6	108	5238	42*
g33896 × 7	126	7240	49*
g33918 × 4	72	2267	28*
g33918 × 5	90	3629	35*
g33918 × 6	108	5309	42*
g33918 × 7	126	7303	49*
g33920 × 4	76	2491	32*
g33920 × 5	95	4019	40*
g33920 × 6	114	5860	48*
g50634 × 3	60	1711	30*
g50634 × 4	80	3066	40*
g50634 × 5	100	4823	50*
g50634 × 6	120	6966	60*
1dc.64-c	64	1473	10*
1dc.128-c	128	6657	16*
2dc.128-c	128	2955	5*
1tc.64-c	64	1824	20*
1tc.128-c	128	7616	38*
1et.64-c	64	1752	18*
1et.128-c	128	7456	28*
1zc.128-c	128	5888	18*
chv12 × 5	60	1525	10*
chv12 × 6	72	2247	12*
chv12 × 7	84	3124	14*
chv12 × 8	96	4147	16*
chv12 × 9	108	5281	18*
myc5 × 12	60	1674	24*
myc5 × 16	80	3032	32*
myc5 × 20	100	4747	40*
myc11 × 5	55	1288	10*
myc11 × 7	77	2624	14*
myc11 × 9	99	4442	18*
myc23 × 3	69	1759	6*
myc23 × 4	92	3403	8*
myc23 × 5	115	5539	10*
s3m25 × 2	50	910	8*
s3m25 × 3	75	2290	12*
s3m25 × 4	100	4274	16*
s3m25 × 5	125	6892	20*
C125.9	125	6963	34*
hamming6-2	64	1824	32*
hamming6-4	64	704	4*
johnson8-4-4	70	1855	14*
johnson16-2-4	120	5460	8*

quantum computers continues to increase, the problems in this benchmark can be quite easily scaled up as well to keep pace with the hardware technology.

Classical algorithms for MCP

In the present section, we focus on results for the clique problem arising from 2014 that were based on previous research and those that extend the classical results. First, we detail the algorithmic landscape of the exact solvers and their background. Second, we will shortly introduce some recent results in the heuristic approach.

Exact algorithms

From the historical point of view, classical algorithms, i.e., exact and heuristic algorithms, for solving the MCP have primarily focused on branch-and-bound approaches[113], or, in other terms, backtracking, with several significant advancements in recent years. Previously, more general algorithms were known, such as the Bron–Kerbosch backtrack algorithm[114], which enumerated all maximal cliques and thus solved the MCP as a byproduct. Later, specific algorithms tailored for the MCP were developed. One of the early and influential algorithms for MCP was introduced by Carraghan and Pardalos[115]. This algorithm, known for its simplicity and effectiveness, employed a branch-and-bound method and was a good base to add pruning strategies in the following algorithms to reduce the search space. The algorithm performs a depth-first search from each vertex and prunes the search space by considering the number of remaining vertices that could potentially form a clique.

In the early years, several basic ideas of auxiliary algorithms aiding the solution of the MCP were explored. An early algorithm from Balas and Yu[116] was mostly unnoticed due to its complexity, but contained many ideas that are very important nowadays—for example, the usage of coloring the nodes of the graph. From recent years, we can mention an algorithm by Konc and Janežič[117] that explores the usefulness of the concept of reordering the nodes, and another from Tomita[118] that takes a more conservative approach. An especially interesting approach of the Russian doll technique was presented by Östergård[119]. Finally, would like to mention probably the first article about parallelization of the MCP problem[120].

A comprehensive overview covering classical methods prior to 2014 can be found in ref. 45. Here we shall further refer to the more recent papers.

Auxiliary algorithms. We shall describe the branch-and-bound algorithms later; here, we would like to point out an important phenomenon concerning these. The running time of these algorithms is naturally closely connected to the search tree size. One factor is of utmost importance for the size of the search tree, and that is the branching factor[121]. In the case of the state-of-the-art MCP solvers, this branching factor depends on the actual lower and upper bounds. The first is usually an incumbent solution, while the latter is obtained by an auxiliary algorithm. The most widely used is the coloring of the nodes of the graph. The bigger the gap between lower and upper bounds usually the bigger

the branching factor and thus the size of the search tree. Thus, modern clique search algorithms pursue two goals: find a good incumbent solution fast and produce a tight upper bound during the search.

A coloring of the nodes of the graph $G = (\mathcal{V}, \mathcal{E})$ can be conveniently given by a map $f : \mathcal{V} \to \{1, \ldots, c\}$. Here, the numbers $1, \ldots, c$ represent the colors and $f(v)$ is the color of the node $v \in \mathcal{V}$. The coloring of the nodes of a graph is called proper or legal coloring, and has two properties. First, each node receives exactly one color. Second, two nodes of the same edge cannot receive the same color. The minimum number of colors that a graph can be colored is called the chromatic number and is denoted by $\chi(G)$. Naturally, $\chi(G) \geq \omega(G)$, as for a clique of size k, one needs k different colors for proper coloring. A greedy coloring is an upper bound to $\chi(G)$ and thus also an upper bound to the maximum clique size. The coloring may be used in the very beginning of the algorithm for the whole graph, or it can be performed at each node for a subgraph. Later takes more time, but has a stronger effect on the size of the search tree. The gain and balance between using coloring only on top or at each level, and also the complexity of the used coloring algorithm, including simplified partial coloring or recoloring using previous coloring results, is actively researched.

Such an upper bound proved so efficient that many state-of-the-art solvers use coloring or even stronger bounds on each level of search. Among others, these may include more complex coloring schemes, such as b-fold coloring, fractional coloring, and the Lovász theta function. For a detailed list and comparison of different upper bounds in clique search, see Margenov et al.[122].

Another set of auxiliary algorithms, referred to as reduction rules, is used mainly for preprocessing purposes in order to decrease the size of the graph and/or complexity of the problem. These can be seen as a polynomial time transformation of $G = (\mathcal{V}, \mathcal{E})$ into $G' = (\mathcal{V}', \mathcal{E}')$ such that G' has special properties. Mostly, these transformations fall into two categories:

1. $|V'| = |V| - 1, \omega(G') = \omega(G)$
2. $|V'| \leq |V|, \omega(G') = \omega(G) - 1$

The first category is about a transformation where we can delete a node from the graph and preserve the clique number. An example of such a transformation is the dominance of two nodes. A node $u \in G$ can be deleted from the graph if there is a node v present, such that $\partial u \subseteq \partial v$.

The second category is about a transformation where the size of the graph is reduced while the clique number is also reduced by one. An example of that is the so-called vertex folding transformation. If a node v has degree $N - 3$, that is, only two nodes u, w are non-neighbors to it, and they are adjacent, that is $\{u, w\} \in \mathcal{E}$, then one can remove v, u, w, and replace them by folding u, v into v', which has the common neighborhood of u, w. (We would like to point out that vertex folding, among several other transformation methods, is a special case, and thus a derivative of the structure transformation[123,124]).

Similar transformations are widely used for fixed-parameter tractable (FPT) algorithms and called kernelization in that field. The MCP does not admit an FPT algorithm, as it was detailed in the previous section, but the kernelization methods can be useful for problem reduction. The vast landscape of FPT algorithms and kernelization is out of the scope of this paper; we only refer to some previous works that are more connected to the underlying MCP[125–130].

It is important to note that branch-and-reduce algorithms, built upon kernelization methods, have proved to be extremely successful for the Vertex Cover problem. Moreover, the usual test graphs for the VCP are very different from the test graphs for the MCP. The hard instances for MCP usually have 500–1000 nodes and a density of 50−90%, while the instances for VCP—which should be considered as complement graphs to the instances for MCP—have several thousands to even millions of nodes, with extremely small densities. Some researchers, such as those in ref. 129, have indicated that these graphs differ significantly, leading to the conclusion that different algorithmic approaches are required for them. Specifically, kernelization is suitable for the typical VCP, whereas branch-and-bound is appropriate for the typical MCP. However, the PACE2019 competition demonstrated that these approaches could be successfully combined; after applying strong kernelization, the kernel of the problem should then be addressed using the branch-and-bound method. All three medalists employed this strategy[101]. Note that the problem setup of very spars graphs where the MCP is hard upon reason of the graph being huge is a valid problem, and there are notable results in this area[131–134]. But in our paper, we focus more on the small and hard instances.

Obviously, the presented two categories of transformations can be extended. In the first case, instead of deleting a node, one can aim to delete an edge, that is, a transformation yielding to $|\mathcal{E}'| = |\mathcal{E}| - 1$, as discussed in detail in Szabó et al.[27]. In the second case, one can propose increasing transformations, that is $|\mathcal{V}'| \geq |\mathcal{V}|$. Although they cannot be called reductions, they can still result in either an easier problem or transform the graph into a bigger one that can be tackled by future reductions more efficiently, as described in Gellner et al.[134]. It is clear that this question requires more thorough research, and it is very possible that future state-of-the-art programs will include such reductions.

Current achievements in exact algorithms with branch-and-bound approaches. Exact algorithms have played a pivotal role in advancing our ability to solve the maximum clique problem (MCP). Over the years, several methodologies have been proposed, refined, and optimized. From 2014 onwards, the field has seen significant advancements, particularly in the areas of branch-and-bound techniques. Branch-and-bound methods have been at the core of most exact algorithms for solving the MCP. These algorithms explore the solution space by branching on vertices and bounding using stronger bounds, thereby systematically excluding some parts of the solution space that do not contain an optimal solution.

A branch-and-bound algorithm for the MCP typically starts by selecting a vertex, treating it as a single-vertex clique, and attempts to extend this clique by adding one vertex at a time from the remaining vertices. If the addition of a vertex results in a larger clique, the algorithm continues to explore this branch. However, if there are no more vertices to add, the algorithm backtracks and explores other branches. The bounding part of the algorithm comes from the ability to estimate an upper bound on the size of the maximum clique that can be obtained from a given branch. If this upper bound is smaller than the size of the largest clique found so far, this branch can be pruned. We refer to refs. 45,113,115 for an in-depth discussion on branch-and-bound approaches.

San Segundo and others, in a series of papers from 2015 and 2016, enhanced their previous bit-parallel BB-MaxClique (BBMC) algorithm using several innovative techniques. While most of the ideas seem small, the overall result accumulates to a much better algorithm. They describe a technique of producing an upper bound that is actually better than the upper bound derived from the chromatic number. They call this bound Infra-Chromatic[135], which was improved in San Segundo et al.[136] and enhanced in ref. 137. They combined this technique and thoroughly examined the effect with different vertex ordering strategies[138], and also introduced a very effective method of bit encoding for the coloring subprogram for efficient and fast chromatic bound search for the branches[139].

Based on previous results from 2010[140] Chu-Min Li, Hua Jiang, and others[141] proposed for the MCP a strategy named Large MaxClique (LMC) for the branch-and-bound methods has been established. They combined a preprocessing strategy to delete nodes that cannot belong to a maximum clique with the incremental MaxSAT reasoning for pruning branches. In ref. 90, they extend these results by applying node ordering. The static vertex ordering in G must be kept during the search and dynamic ordering of the vertices compared, and along with a version of mixed strategy, three programs were evaluated (Static ordering MaxClique Solver—SoMC, Dynamic ordering MaxClique Solver—DoMC, Mixed ordering MaxClique Solver—MoMC). In their next paper[142], they investigate the result of searching for a large clique early, an incumbent solution, if that helps the algorithm (incremental upper bound MaxClique solver – IncMC2). All proposed versions were superior in many cases to the other approaches at that time on random and DIMACS instances as well.

An algorithm presented by Szabo and Zavalnij[143] uses several smaller contributions that add up to an algorithm, which is comparable to other state-of-the-art programs, sometimes slower, and for some instances still better than others up to date. The key points of the algorithm are the following. First, they use the k-clique decision problem instead of MCP because that leads to better preconditioning and better branch factor. The MCP is solved by performing several k-clique search in an incremental manner. Second, they use a different reordering of the nodes in the branch opposite to other approaches, as the goal is not to find a better incumbent solution but to reduce the search tree, see also Ouyang[144]. Third, they use a strong coloring scheme, incremental recoloring in the beginning, so to have the possibly best coloring for the graph. They also perform recoloring of nodes in each branch. Finally, they used an approach of constructing the branching node set described in detail in Szabó et al.[145].

Recently, in San Segundo et al.[103], an exact algorithm, called CliSAT, to solve the MCP to proven optimality has been presented. This method is a combinatorial branch-and-bound algorithm, leveraging an advanced branching scheme enriched with two bounding strategies to minimize the branching tree size. The initial strategy employs graph coloring procedures and partial maximum satisfiability problems arising in the branching scheme. The second strategy incorporates a filtering stage utilizing constraint programming and domain propagation methods. CliSAT, tailored for structured and computationally challenging MCP instances, excels in dense environments characterized by numerous, interconnected large cliques. Rigorous testing on tough benchmark problems and newly identified challenging instances from diverse fields reveals CliSAT's superior performance over leading MCP algorithms, sometimes by several orders of magnitude. Specifically devised for demanding MCP problems with tens of thousands of vertices, these instances are marked by dense structures and multiple large, intertwined cliques. In this context, cutting-edge techniques are combinatorial branch-and-bound algorithms with bounding processes based on graph coloring and partial maximum satisfiability issues emerging from the branching strategy. CliSAT demonstrates remarkable results on the classical DIMACS benchmark, surpassing previous top-performing exact algorithms. Furthermore, in several categories of challenging instances, CliSAT achieves unparalleled performance, surpassing others significantly. CliSAT also shows competitive results for sparse graphs with as many as 150000 vertices, despite not being primarily designed for such instances.

CliSAT utilizes a constructive, n-ary branching approach, progressively constructing a clique by adding vertices one by one in a recursive manner. Within this context, $\widehat{C} \subseteq \mathcal{V}(G)$ represents the clique linked to a specific branching node, comprising vertices included in $\widehat{C}$ during previous branching steps. Additionally, each branching node corresponds with a subproblem graph G', encompassing vertices that can be individually added to $\widehat{C}$. Throughout its operation, CliSAT monitors the current best solution, referred to as C_{inc}. The lower bound, denoted as lb, is the size $|C_{inc}|$ of this solution. When a larger clique is identified during branching, signified by $|\widehat{C}| > lb$, both C_{inc} and lb are updated. Post execution, C_{inc} represents the maximal clique in G. The main idea of the branching scheme is to partition the vertex set of the subproblem graph G' into two groups: (i) the branching set B and (ii) the pruned set P—based on refs. 122,145.

Given the definition of P, at least one vertex from $B = \mathcal{V}(G')\backslash P$ is required to enhance the incumbent solution C_{inc}. Consequently, branching on vertices in P at a certain node is unnecessary, leading the algorithm to backtrack when B becomes empty. Once the pruned and branching sets are established, CliSAT executes a $|B|$-ary branching, generating a new branching node for each vertex in B by including it in the current subproblem clique $\widehat{C}$. They also use strong incremental upper bounds for constructing the set P as discussed in previous works, like ref. 135,142,146.

Although scarce and considered extremely hard for combinatorial optimization, some examples of parallel algorithms were also published. Depolli et al.[147] used shared memory parallelization up to 24 cores, while Zavalnij[148] used distributed parallelization up to 512 cores. Finally, we would like to mention an interesting and unpublished concept by Hrga[149] presented on the Zuze UG Workshop, which is apart from demonstrating a parallel solver for the MCP, also using the Lovasz' Theta function as a bounding function. Note, though, that the effective parallelization of the clique problem still remains mostly open for future research.

Recent years have seen the development of precise algorithms for variations and extensions of the MCP. Papers[150–152] have detailed efficient algorithms for the maximum vertex-weighted clique problem. In this problem, one searches for a clique, where the sum of the weights on the nodes of the clique is maximal. The edge-weighted version of the problem has been addressed in works by San Segundo et al.[153] and Shimizu et al.[154]. A special case of the weighted problem, namely searching for maximum weight k-cliques in k-partite graphs, arises from research on the protein docking problem[9]. The authors in Furini et al.[155] have introduced exact algorithms for vertex and edge interdiction variants of the MCP. Lastly, an innovative exact algorithm for the knapsack problem with conflicts, which is akin to the MCP with an added knapsack constraint, was presented in Coniglio et al.[156]. Recent research has also focused on identifying the clique number in large, sparse graphs typical of social networks. Algorithms tailored to these scale-free graphs, characterized by a power-law degree distribution, have been developed. These methods successfully solve the MCP with proven optimality in networks containing millions of vertices, as demonstrated in studies like Hespe et al.[157] and Walteros et al.[158]. However, the techniques effective for these large, sparse graphs are generally not suitable for solving dense and challenging MCP instances.

To summarize the nuts and bolts of making an effective exact MCP solver, we can highlight several key points. First, the effective combinatorial search was implemented as a simpler backtrack, and later improved by using the branch-and-bound technique. Second, better upper bounds were introduced, such as proper coloring of the nodes. This direction of the research was later and still is continued nowadays, with more subchromatic bounds such as Lovasz' Theta function or MaxSAT reasoning. Third, the ordering of the nodes in the branch phase is researched, based on node degree, degeneracy, or color index (number of neighboring color classes). Both approaches, such as finding a big clique fast that is using decreasing order in case of node degrees, and proving the maximum clique size that is using increasing order, are actively researched. Fourth—connected to the previously raised dichotomy—the question of finding a good incumbent solution is raised. Fifth, the active use of preconditioning (kernelization) methods coming from the FPT research community, as well as using such reductions during the search phase itself, seems to play a major role in nowadays state-of-the-art algorithms.

Heuristics for hidden cliques

Spectral and message passing algorithms. The discovery of hidden cliques in an Erdős-Rényi graph G of order N and probability p traces its origins to Kučera's seminal research[159]. Kučera identified that, for sufficiently large constant c, a clique of size $c\sqrt{N \log N}$ will predominantly include the vertices with the most connections. This insight was further advanced by Alon et al.[59], who applied spectral methods. Their study established that the optimal detection of a hidden clique in graph G is feasible when the clique size $|C_{HC}|$is at least $c\sqrt{N}$, with c being a sufficiently large constant.

The approach proposed by Alon (and colleagues) involves the following process. They start with the adjacency matrix A of the graph G, sized $N \times N$, but with entries of ± 1 rather than the typical 0, 1 (i.e., 0 is replaced with -1, while 1 remains unchanged). In scenarios where graph G is generated according to the Erdős-Rényi model with parameters N and edge-formation probability $p = 0.5$, it is improbable for matrix A to have an eigenvalue exceeding $2\sqrt{N}$. However, if there is a clique C_{HC} within G, the adjusted matrix A will exhibit an eigenvalue of at least $|C_{HC}| - 1$. Thus, if $|C_{HC}|$ is significantly greater than $2\sqrt{N}$, the largest eigenvalue of A becomes a reliable indicator to differentiate between Erdős-Rényi graphs and those containing hidden cliques. The eigenvector linked to this largest eigenvalue can then be utilized as an initial step in pinpointing the actual hidden clique C, although additional efforts are necessary to precisely locate it. Alon

observed that the constant in the expression $|C_{HC}| = c\sqrt{N}$ can be optimized. By randomly selecting a fixed number s of vertices from the graph G, there is approximately a $(|C|/|C_{HC}|)^s$ chance that these vertices form part of a clique C. Therefore, we can expect that after about $N^{s/2}$ tries, s vertices from C will be chosen. Focusing next on the subgraph G', formed by the mutual neighbors of these vertices, it likely contains just $N/2^s$ vertices. This subgraph remains random and includes a nearly $c2^{s/2}\sqrt{N/2^s}$ size clique. Consequently, in the context of G', solving a hidden clique problem becomes more feasible with these improved parameters. This leads to the implication that for any $\epsilon > 0$, it is possible to identify planted cliques of size $\epsilon\sqrt{N}$ in a time complexity of $O(N^{\log 1/\epsilon})$[160].

The leading approach for uncovering hidden cliques in random graphs (Erdős-Rényi), backed by theoretical guarantees, is credited to the algorithm proposed by Deshpande et al.[61]. Their analysis, which becomes asymptotically exact as N approaches infinity, demonstrates the algorithm's high probability of detecting hidden cliques of size $|C_{HC}| \geq \sqrt{N/e}$. This algorithm is an extension of belief propagation (BP), a heuristic approach in machine learning for estimating posterior probabilities in graphical models. BP has also been studied in statistical physics[161–173]. As stated in refs. 174–176, BP calculates marginal probabilities for individual nodes within a factor graph. It delivers precise results on tree structures and has also proven effective on graphs with loops, as indicated in refs. 61,177–179. As an iterative method, BP exchanges messages between links and nodes to ascertain marginal probabilities for each node. Once BP converges, indicating that marginal probabilities are determined, it sorts nodes based on these probabilities to solve the problem. Nonetheless, BP may fail or converge to a non-informative fixed point if the graph does not resemble a local tree. In BP's implementation for graphical models, each variable node represents a site in the original graph $G(\mathcal{V}, \mathcal{E})$, while function nodes correspond to links in $G(\mathcal{V}, \mathcal{E})$. For an in-depth understanding of the algorithm by Deshpande et al.[61], readers are directed to their comprehensive paper. Here, we just recall the main steps.

The algorithm operates on a fully connected graph, characterized by an adjacency matrix A with entries of ± 1, similar to the spectral method introduced above. It functions through an iterative process, sending messages from links to nodes and then calculating specific values for each node. These values indicate whether a node is part of the hidden clique. This algorithm strikes a balance in terms of complexity and computational demands, positioning itself between simpler local methods, like greedy search approaches, and more comprehensive global strategies, such as the spectral techniques mentioned above. For our survey, we adapted a straightforward version of the algorithm from Deshpande et al.[61], employing the equations they outlined. We briefly revisit these equations here:

$$\begin{aligned}\Gamma_{i\to j}^{t+1} &= \log\frac{|C_{HC}|}{\sqrt{N}} + \sum_{l\neq i,j}^{N}\log\left(1+\frac{(1+A_{l,i})e^{\Gamma_{l\to i}^{t}}}{\sqrt{N}}\right)\\ &\quad -\log\left(1+\frac{e^{\Gamma_{l\to i}^{t}}}{\sqrt{N}}\right),\end{aligned} \tag{5}$$

$$\begin{aligned}\Gamma_{i}^{t+1} &= \log\frac{|C_{HC}|}{\sqrt{N}} + \sum_{l\neq i}^{N}\log\left(1+\frac{(1+A_{l,i})e^{\Gamma_{l\to i}^{t}}}{\sqrt{N}}\right)\\ &\quad -\log\left(1+\frac{e^{\Gamma_{l\to i}^{t}}}{\sqrt{N}}\right),\end{aligned} \tag{6}$$

where $\Gamma_{i\to j}^{t}, \Gamma_{j\to i}^{t}, \Gamma_{i}^{t} \in \mathbb{R}_{+}$.

Equations (5) and (6) detail the dynamics of message exchanges and the computation of vertex parameters Γ_i^t in a fully connected graph. This is facilitated by the adjacency matrix A, which accounts for both the presence and absence of links between nodes (± 1, respectively). Logarithms are employed in these equations for enhanced numerical stability. The initial conditions for the messages in (5) are randomly assigned values less than zero.

The significance of $\sqrt{N}$ is noted as a critical scaling factor for problems involving hidden cliques.

Equation (5) is focused on updating the message sent from node i to node j. This update is derived from all incoming messages to node i from the previous iteration, excluding the message from node j to node i. These messages are represented as odds ratios in real numbers $\mathbb{R}$, indicating the likelihood of node i being part of the hidden set C_{HC}. The message from node i to node j conveys whether node i is likely part of C_{HC}, based on the odds ratios from the other $N-2$ nodes in the graph, excluding nodes i and j. A positive logarithmic difference in the sum suggests a higher probability of a node being in C_{HC}, while a negative difference implies a lower likelihood.

Equation (6) deals with the numerical update of the vertex parameter Γ_i^t, calculated from all incoming messages to node i. It estimates the probability of node i being in the hidden clique C_{HC}. Nodes more likely to be in the hidden clique will have a $\Gamma_{i\in C_{HC}}^{t_c}$ value greater than zero, while those unlikely to be in C will have a $\Gamma_{i\notin C_{HC}}^{t_c}$ value less than zero. t_c is the time when the convergence is achieved. As iterative BP equations (5) and (6) are useful only if they converge. The computational complexity of each iteration is $\mathcal{O}(N^2)$, equation (5) can be computed efficiently using the following observation:

$$\Gamma_{i\to j}^{t+1} = \Gamma_{i}^{t+1} - \log\left(1+\frac{(1+A_{j,i})e^{\Gamma_{j\to i}^{t}}}{\sqrt{N}}\right) + \log\left(1+\frac{e^{\Gamma_{j\to i}^{t}}}{\sqrt{N}}\right). \tag{7}$$

The convergence of all messages and vertex quantities in the algorithm typically requires $\mathcal{O}(\log N)$ iterations. As a result, the overall computational complexity of the algorithm is $\mathcal{O}(N^2 \log N)$. When the messages in equation (5) reach convergence, the vertex quantities computed by equation (6) are arranged in descending order. The top h_{HC} elements are then selected for verification as a potential solution. If these elements constitute a solution, the algorithm concludes successfully; otherwise, it reports a failure. While this algorithm is theoretically exact as N approaches infinity, its efficacy has been empirically validated on Erdős-Rényi random graphs with planted solutions. In these tests, a hidden clique of size $C_{HC} \geq \sqrt{N/e}$ was embedded, and the numerical outcomes were consistent with the theoretical predictions.

Monte Carlo methods. The application of Monte Carlo methods to the hidden clique problem represents an intriguing approach in computational graph theory. While Monte Carlo methods have been extensively used in various domains for probabilistic and numerical analysis, their application in detecting planted cliques within graphs is relatively recent. Jerrum[58] highlighted a significant limitation of Monte Carlo methods (MCM) in this context, demonstrating that these methods cannot efficiently uncover hidden cliques of size $O(N^{0.5-\epsilon})$ with $\epsilon > 0$. This finding initially cast doubt on the efficacy of MCM, especially for cliques smaller than the square root of the graph's size. However, recent advancements have rekindled interest in this area. Both numerical and theoretical analyses have shown interesting results. From some observations in Montanari[180], Angelini[181] presented an accurate analysis on the performance of parallel tempering for the HCP. Parallel tempering[182], also known as replica exchange Monte Carlo, is a sophisticated computational technique used primarily in statistical mechanics and computational chemistry. The algorithm enhances the sampling efficiency of Monte Carlo simulations, especially in systems that experience slow dynamics due to energy barriers. The core concept of parallel tempering involves running multiple simulations (or replicas) concurrently at different temperatures. The idea of this method is to make configurations at high temperatures available to the simulations at low temperatures and vice versa. In her work, Angelini[181] applies parallel tempering to HCP, showing numerically that its time-scaling in the hard region is indeed polynomial for the analyzed sizes.

From a theoretical point of view, Gamarnik et al.[183] examined the landscape of sufficiently dense subgraphs of G and their overlap with the planted clique. Employing the first-moment method, they provided evidence of a phase transition for the presence of the Overlap Gap Property (OGP)[184] when $|C_{HC}|$ is $\Theta(\sqrt{N})$. They presented the first concentration

results for the $|C_{HC}|$-densest subgraph problem in the Erdős-Rényi random graph $G(N, p = 0.5)$, when $|C_{HC}| = N^{0.5-\epsilon}$, with $\epsilon > 0$. Following this observation, Angelini et al.[185] conducted an exhaustive numerical analysis demonstrating that a Monte Carlo algorithm of the same class used by Jerrum[58] is generally suboptimal for recovering the planted clique. Specifically, they found that a Metropolis MC based on the posterior distribution is suboptimal in identifying a planted clique; such an algorithm does not succeed in polynomial time even when $|C_{HC}|$ is significantly larger than $O(\sqrt{N})$. However, they noted that a Parallel Tempering algorithm, when generalized by adding an inverse temperature parameter $\beta = 1/T$ with values $\beta \in (0, \infty)$, can match the performance of Bayes optimal algorithms such as BP, effective in polynomial time down to $|C_{HC}| = \sqrt{N/e}$. More precisely, when the inverse of temperature $\beta \neq 1$, the Parallel Tempering algorithm achieves the performance level of BP, whereas with $\beta = 1$, the algorithm becomes suboptimal. This is attributed to the fact that MC algorithms work better at higher temperatures, where the global minimum of the free energy remains the planted solution, but entropic effects facilitate faster movement through the phase space, allowing the solution to be reached in less time.

These observations, namely those of Gamarnik et al.[183] and Angelini et al.[185], were corroborated in a recent publication by Chen et al.[186]. In this study, Chen et al.[186] revisited the original Metropolis algorithm proposed by Jerrum[58]. They demonstrated that, contrary to other efficient algorithms which succeed when $\eta = 0.5$ (as seen in SM^1-ES[187]), the Metropolis algorithm fails to recover a planted clique of size $|C_{HC}| = \Theta(N^\eta)$ for any constant $0 \leq \eta < 1$. The authors reinterpreted Jerrum's results, noting that Jerrum's findings indicate the existence of a starting state (instance-dependent) where the Metropolis algorithm does not find the planted clique within polynomial time. Furthermore, they argue that the Metropolis algorithm is ineffective over a broad range of temperatures when initiated from the most natural starting point–the empty clique. This addresses an open question posed in Jerrum's original paper[58]. The paper also concludes that a simulated tempering variant of the Metropolis algorithm is equally ineffective in the same parameter regime. In contrast to these results, a recent pre-print by Gheissari et al.[188] demonstrates that a simple Markov Chain Monte Carlo (MCMC) approach, utilizing an energy function that relaxes that of Jerrum's Metropolis process, can effectively find the planted clique down to the $|C_{HC}| = \sqrt{N}$ threshold in $O(N)$ steps, starting from a natural uninformed state. To achieve this, they relaxed the clique constraint for this process. This adaptation allows the Markov chain to evolve across all subsets of the full vertex set while incorporating a penalty term in the Hamiltonian for any missing edges. Moreover, they show that when initialized from the full vertex set, both the gradient descent and its low-temperature MCMC analog recover the planted clique in linearly many steps, as long as $|C_{HC}| = \Omega(\sqrt{N})$. It's important to note that by relaxing the problem to the space of subgraphs, they introduce two intuitive uninformed starting points for the algorithm: the empty set or the full graph. Crucially, the selection of the starting point plays a significant role in the algorithm's success. Similar to the Metropolis process focused on cliques, they observe that starting with an empty set renders gradient descent ineffective in locating the planted clique.

Learning algorithms for MCP: the graph neural networks (GNNs)

In recent decades, Artificial Intelligence (AI) has been a game-changer in various fields, thanks to its ability to process complex tasks and handle large datasets. Machine learning, a critical branch of AI, features algorithms that learn and make decisions or predictions from data. A more advanced subset of machine learning is Deep Learning (DL), renowned for its effectiveness in building and training multi-layered neural networks. DL has achieved notable success in areas dealing with extensive and intricate datasets[189–192]. One of its landmark achievements is AlphaFold[193], developed by DeepMind, which has made significant strides in solving the protein folding problem–a longstanding challenge in biology and in the realm of combinatorial optimization. Building upon recent advancements in DL, Graph Neural Networks (GNNs) have emerged as a groundbreaking tool in addressing combinatorial optimization challenges[194]. GNNs, uniquely structured to process graph-formatted data, are adept at making predictions related to nodes, edges, and various graph-centric tasks. Each GNN layer processes node embeddings and the adjacency matrix to produce updated node embeddings. This innovative architecture equips GNNs with the capability to efficiently tackle combinatorial issues, where the overall quality of a solution is often contingent on the collective configuration rather than isolated elements. Encouragingly, GNNs can be Turing universal in the limit[195], which motivates their use as general-purpose solvers.

The core idea behind GNNs is to learn how to incorporate the connections and features of both nodes and edges to make predictions or classifications. In these networks, each node gathers information from its neighbors through a process called message passing. During each iteration of this process, nodes aggregate and transform information from their neighbors, gradually updating their own state. The final node representations can then be used for various graph-level, node-level, or edge-level tasks.

A GNN can be conceptually viewed as comprising layers, each represented by a function (activation function) $\mathbf{F}[\,\cdot\,]$ with parameters Φ. This function processes node embeddings and the adjacency matrix to produce updated node embeddings. The network's operations can be expressed as:

$$\begin{aligned} \mathbf{H}_1 &= \mathbf{F}[\mathbf{X}, A, \phi_0] \\ \mathbf{H}_2 &= \mathbf{F}[\mathbf{H}_1, A, \phi_1] \\ \vdots &= \vdots \\ \mathbf{H}_K &= \mathbf{F}[\mathbf{H}_{K-1}, A, \phi_{K-1}] \\ f[\mathbf{X}, A, \Phi] &= \mathbf{F}[\mathbf{H}_K \mathbf{1}/N]. \end{aligned} \tag{8}$$

Here, $\mathbf{X}$ denotes the input matrix (i.e., the vertex embedding matrix), A is the adjacency matrix underlying the graph, $\mathbf{H}_k$ encompasses the evolved node embeddings at the kth layer, and ϕ_k (for $k = 0, \ldots, K-1$) represents the parameters of each respective layer. The function $f[\mathbf{X}, A, \Phi]$ signifies the network's output, serving the learning objective, be it classification or regression. This output function is instrumental in defining the loss function for both supervised and unsupervised learning settings. For comprehensive insights on GNNs, references such as refs. 196,197 offer detailed discussions.

In the realm of combinatorial optimization, GNNs demonstrate a remarkable ability to either forecast the qualities of various configurations or directly deduce solutions[198–206]. They undergo training using a series of problem instances with established optimal solutions. During this training phase, GNNs discern patterns and structures, enabling them to apply these insights to novel, unencountered instances. This learning mechanism positions GNNs as a powerful instrument in the field of optimization, offering a blend of precision and adaptability. For an introduction to the field of GNNs for combinatorial optimization, we refer to Section 8.1.6 of ref. 196, here we review the main papers regarding MCP.

An unsupervised approach for MCP has been discussed in Karalias and Loukas[207]. In their paper, the authors introduce an unsupervised learning model that innovatively constructs a differentiable loss function. This function's minima are assured to yield low-cost, valid solutions to the problem at hand. Inspired by Erdős's probabilistic method, this approach trains a GNN to generate a distribution over subsets of an input graph's nodes by minimizing a probabilistic penalty loss function. Crucially, this achievement is realized in a straightforward, mathematically grounded manner, and such a loss acts as a certificate for the existence of a low-cost set[207]. It eschews the need for continuous relaxations, regularization, or heuristic adjustments to correct improper solutions. Successfully optimizing this loss function ensures the derivation of high-quality integral solutions that conform to the problem's constraints. Upon training completion, the authors apply a well-established technique from the realm of randomized algorithms. This technique is used to sequentially and deterministically extract a valid solution from the learned distribution. This method eliminates the need for polynomial-time reductions, directly addressing each

problem instead. Experiments for MCP were conducted on the IMDB, COLLAB[107,208] and TWITTER[108] datasets. Further experiments were done on graphs generated from the RB model[91], which has been specifically designed to generate challenging problem instances. The GNN shows high performance on such datasets.

Another innovative unsupervised approach is introduced in Min et al.[209]. This paper's objective is to utilize a geometric scattering-based GNN for efficiently approximating the solution to the MCP. The model stands out for its parameter efficiency, requiring only approximately 0.1% of the number of parameters compared to earlier GNN baseline models. The method comprises three primary components. The first is a hybrid scattering-GNN model, which converts a set of basic node-level statistics into a probability vector $\vec{p} \in [0,1]^N$. This vector efficiently represents the likelihood of each node being part of the maximum clique. The graph scattering transform, central to this model, utilizes a wavelet-based approach for machine learning on graphs, as discussed in refs. 210–212. Different from the one-hop localized low-pass filters in certain GNNs that promote smoothness among neighboring nodes, these wavelets function as band-pass filters. They encapsulate long-range dependencies through the expansive spatial support of their aggregations. Conceptually, at each node, diffusion wavelets perform a comparative operation, assessing the difference between the averaged features of two differently sized neighborhoods. The second component is a constraint-preserving decoder, designed to identify the maximum clique. The ideal output for a graph G with N nodes is a representation $\vec{p} \in [0,1]^N$, where nodes within the maximum clique are assigned a value of 1, while all others receive 0. This allows for the extraction of the maximum clique using $O(N)$ operations. In practice, the model constructs a representation that assigns higher probabilities to nodes within the maximum clique and lower probabilities to others, facilitating the extraction of the maximum clique by selecting nodes with higher p_i values. Experiments conducted on the IMDB, COLLAB, and TWITTER datasets demonstrate the high performance of the GNN, albeit with similar efficiency compared to the algorithm presented in Karalias et al.[207].

The burgeoning debate in the field of artificial intelligence has recently raised the question of whether GNNs can rise to the challenge and supersede the performance of state-of-the-art classical algorithms, such as greedy or message-passing strategies. An intriguing development in this discourse is the recent publication of the article[213]. In this study, the authors explore the application of GNNs in addressing combinatorial optimization challenges. The core of their approach involves defining a Hamiltonian (cost function) H, representative of the optimization problem. This function is characterized by binary decision variables $x_i \in \{0,1\}$ $(i = 1, \ldots, |\mathcal{V}|)$, each linked to a corresponding vertex i within an undirected graph $G(\mathcal{V}, \mathcal{E})$. The graph structure effectively captures the interdependencies among these variables. To facilitate the use of GNNs, they employ a relaxation technique on the problem Hamiltonian, creating a differentiable loss function. This function is pivotal in training the GNN. Adhering to a recursive neighborhood aggregation framework, the GNN iteratively updates the feature vector of each node based on the aggregated information from its immediate neighbors. Upon completion of the unsupervised training phase, a projection heuristic is applied. This step is crucial for translating the continuous-valued outputs, denoted as p_i, back into binary decisions, $x_i \in \{0, 1\}$. They demonstrate the efficacy of their methodology through numerical experiments focused on the MIS and Max-Cut problems, showcasing promising results. In this study, the authors claim, *"We find that the graph neural network optimizer performs on par or outperforms existing solvers, with the ability to scale beyond the state-of-the-art to problems with millions of variables"*[213]. This assertion has sparked a flurry of responses in the AI community.

The initial critique of the work presented in Schuetz et al.[213] was put forth by Stefan Boettcher[214]. Boettcher's analysis reveals that, upon detailed examination, the performance improvements of the GNN described in the original study are marginally better than those achieved through gradient descent. This GNN is outperformed by a greedy algorithm, particularly in the context of solving the Max-Cut problem. Similarly, the commentary by Angelini and Ricci-Tersenghi[215] highlights another perspective. They demonstrate that a relatively simple greedy algorithm, which operates in near-linear time, can consistently produce higher-quality solutions for the MIS problem. Furthermore, this is achieved in a significantly shorter timeframe compared to the GNN approach discussed in Schuetz et al.[213]. These critiques collectively suggest that while GNNs hold promise, their current efficacy, especially in comparison to more traditional algorithms like greedy methods, may be limited in certain combinatorial optimization scenarios. Responses from the authors to these comments can be found in refs. 216,217.

The final contribution to this ongoing discussion, including the comments previously mentioned, was provided by Gamarnik[218]. In this manuscript, Gamarnik introduces the OGP as a novel perspective for understanding the performance of GNNs. This work not only adds depth to the existing dialog but also presents a groundbreaking approach in the analysis of GNNs within the context of combinatorial optimization problems. He has asserted that the presence of OGP significantly narrows the possibility for GNNs to surpass existing algorithms. In fact, he elucidates a fundamental constraint for deploying GNNs on random graphs, a constraint that holds true across a diverse spectrum of GNN architectures. This limitation pertains when the GNN's depth remains constant and does not scale proportionally with the size of the graph. This holds true irrespective of other facets of the GNN architecture, such as the internal dimension or update functions. However, it is worth mentioning that this does not exclude GNNs from being a valuable tool in the AI arsenal. While GNNs may face limitations in certain situations, their adaptability and learning potential position them as valuable tools for challenges where traditional algorithms falter. This is particularly true in dynamic environments or with complex, intricate graph data. Therefore, the decision to use classical algorithms or GNNs should focus on utilizing their respective advantages for varying problem types, rather than pursuing a universal solution. This situation highlights the richness of the methodologies available to navigate the complex realm of combinatorial optimization.

Quantum algorithms for MCP

Quantum Computing[219–221] has been proven to be theoretically capable of solving some computational problems in exponentially less time compared to the best-known classical algorithms. Although the exact computational complexity of quantum algorithms with respect to the classical ones is still an open problem, this has not stopped researchers from trying to beat classical algorithms using quantum resources. While it is impossible to be exhaustive about all the different approaches to quantum computing, here we will give a brief introduction to the three main techniques that are used when dealing with combinatorial optimization problems. Such techniques are: Quantum Minimum Finding algorithm (the circuit model)[222], Quantum Annealing[223], and Quantum Approximate Optimization Algorithm (QAOA)[224]. We will see later that all the results obtained in quantum computing regarding the MCP fall into one of these three categories.

Having established the theoretical groundwork and identified the primary quantum computing techniques for tackling combinatorial optimization problems, let's delve into the practical aspects of applying these methods. First of all, we assume to have access to a register of N qubits, i.e., physical objects that exhibit an effective dynamics that can be described by the coherent evolution of a spin 1/2 particle. Each one of the qubits has a state that can be represented in a standard quantum mechanical notation as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where $|0\rangle$ and $|1\rangle$ are the logical states of the system and $\alpha, \beta \in \mathbb{C}$ with $|\alpha|^2 + |\beta|^2 = 1$. The N-qubit register can then be generally described by the state $|\psi\rangle_N = \sum_{i=0}^{2^N-1} \alpha_i |i\rangle$ where $|i\rangle$ is one of logical bit-strings that can be obtained using N bits and again $\alpha_i \in \mathbb{C}$ with $\sum_i |\alpha_i|^2 = 1$. This system lives in a Hilbert space of dimension 2^N and can be manipulated by evolving its state with a carefully designed Hamiltonian H. Indeed, we know that the dynamics of a quantum system follow the Schrödinger equation $i\frac{d}{dt}|\psi\rangle_N = H|\psi\rangle_N$. It means that after a time t, the system will be in the state $|\psi(t)\rangle_N = e^{-itH}|\psi(0)\rangle_N$. Equivalently, we can say that the evolved system, under the application of a Unitary matrix U ($U^\dagger U = \mathbb{I}$, where † is

the adjoint operator and $\mathbb{I}$ the identity matrix), will be $U = e^{-itH}$ and $|\psi(t)\rangle_N = U|\psi(0)\rangle_N$. In the following, we will then talk interchangeably of quantum computations performed by the action of a Hamiltonian H or by the application of a Unitary matrix U onto the state.

Given this theoretical basis, without going into the details of the actual theory of Quantum Computation which is well beyond the scope of this review, it can be proved that with a careful choice of Unitaries U (or Hamiltonians H) we can use the quantum representation $|\psi\rangle_N$ and leverage the power of having a large computational Hilbert space to solve some problems with algorithms faster than the best known classical counterparts. The most famous examples are: factoring of a number N that, on a quantum computer, can be theoretically done in $\sim O(polylog(N))$[225] with respect to the classical $\sim O(N)$; and searching of an element in an unstructured database of N elements that takes $\sim O(\sqrt{N})$ on a quantum computer[226], while $\sim O(N)$ on a classical one.

From a practical standpoint, the endeavor to construct and precisely control a system of N qubits, essential for realizing a Quantum Computer with exceptional capabilities, represents a significant ongoing challenge that is currently a focal point of both academic and industrial efforts. Indeed, there are prototypical quantum devices built from superconducting qubits, trapped ions, neutral atoms, photons, and quantum dots[227] just to cite a few. Unfortunately, none of them has yet reached a stage where error-corrected algorithms can be run reliably and at a scale that achieves a significant advantage over the best classical algorithms. This is due to inherent noise[228,229] and coupling with the environment[230,231] that makes it difficult to scale them up. However, this does not imply that these devices, despite being imperfect and thus falling outside the theoretical guarantees of quantum information theory, cannot serve as effective heuristic solvers for certain problems. Therefore, it is hardly surprising that, despite the technical challenges involved in building a functional quantum computer, some theoretical research has shifted focus towards employing quantum techniques to address some of the most daunting NP-hard problems, including the MCP.

The circuit model

The circuit model of quantum computation is arguably the most popular and widespread model that realizes the capabilities of quantum information processing, described above. In this model of computation, the unitary U that implements the computation is decomposed into combinations of single-qubit operations (or gates) and two-qubit entangling operations. It can be demonstrated that one can approximate with arbitrarily small error any N-qubit unitary by using only these two types of gates[219]. In the circuit model of quantum computation, individual gates are sequentially applied to achieve the desired outcome, mirroring the way classical logic circuits are constructed from a combination of gates. This model is also often referred to as *digital* quantum computation to distinguish it from other *analog* approaches that we will also cover later.

The circuit model algorithm for the MCP is inspired by the Grover search algorithm[226]. Grover's algorithm can find an element by using a quantum circuit that, starting from the uniform superposition state $|\psi\rangle_N = \frac{1}{\sqrt{2^N}}\sum_{i=0}^{2^N-1}|i\rangle$, creates constructive interference in the wavefunction to amplify the probability of measuring a marked state. This state is marked by a so-called Oracle, i.e., a reversible circuit that can verify if a certain state is a solution to our problem or not. The idea of using this Oracle-based model comes from the fact that solutions to NP-hard such as MCP and MIS, are hard to find but can be verified in polynomial time. It would then be theoretically feasible to build a reversible circuit acting in polynomial time that verifies if a state is a solution of MCP, and marks it. One can then use the same logic as Grover's search to find the maximum clique of a graph with $|\mathcal{V}|$ vertices, with a theoretical complexity of $\sim O(|\mathcal{V}|\sqrt{2^{|\mathcal{V}|}})$[111].

It has been demonstrated by subsequent works[232,233] that this algorithm could be implemented in devices using realistic constraints, thus giving a promising outlook. So far, however, no real-world implementation of this algorithm has been put forward due to the limited coherence times of available quantum devices with respect to the required circuit length and the practical cost, in terms of the number of qubits and gates, that is required to actually build an Oracle[232].

Recently, theoretical works have tried to put forward an analog version of this algorithm based on Quantum Walks[234,235]. However, the degree to which these types of algorithms based on Quantum Walks are actually implementable on realistic hardware is still debated, and no experimental results are available.

Quantum annealing

Quantum Annealing[220,236] is one of the most common *analog* forms of quantum computation. It is a computation where the desired final state $|\psi(\tau)\rangle$ is obtained by evolving a suitable initial $|\psi(0)\rangle$ by means of a proper time-dependent Hamiltonian $H(t)$. In Quantum Annealing, the problem to solve is encoded into a spin Hamiltonian:

$$H_P = \sum_i \sigma_i^z + \sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z, \tag{9}$$

where σ_i^z is the Pauli-Z operator acting of qubit i. Finding a solution to this problem means finding the ground state of H_P. To achieve this, we introduce a simpler Hamiltonian $H_I = \sum_i \sigma_i^x$, which possesses an easily preparable ground state. This ground state serves as our initial state, $|\psi(0)\rangle$, from which we proceed to evolve the system using a time-dependent Hamiltonian:

$$H(t) = \left(1 - \frac{t}{\tau}\right)\cdot H_I + \frac{t}{\tau}\cdot H_P, \tag{10}$$

with $t \in [0, \tau]$. Calling $s = t/\tau$ the schedule parameter, (10) becomes:

$$H(s) = (1 - s)\cdot H_I + s\cdot H_P. \tag{11}$$

In order to end up with this protocol in the ground state of H_P, i.e., the correct solution to our problem, we can resort to the adiabatic theorem. Calling $E_0(s)$ and $E_1(s)$ the energies of the ground and first excited states of the Hamiltonian $H(s)$, the adiabatic theorem states that if the system is evolved in a time $1/\tau >> \min_{s\in[0,1]}(E_1(s) - E_0(s))^2$ it will remain in the local ground state of the Hamiltonian, meaning that measuring the resulting state $|\psi(\tau)\rangle$ will give the correct solution[223].

In its general formulation, also called Adiabatic Quantum Computation, this way of performing a quantum computation can be proved to be equivalent to the circuit model. However, with the simple Hamiltonian presented above, it is not, and it can only be used as a heuristic solver for Quadratic Unconstrained Binary Optimization (QUBO) problems. Conveniently, MIS can be easily formulated in QUBO form as shown above.

Indeed, given a graph $G = (V, E)$, the QUBO representation of MIS is

$$H = -A\sum_{i=1}^{N} x_i + B\sum_{(i,j)\in E} x_i x_j, \tag{12}$$

At this point, it is sufficient to map the Hamiltonian (12) onto the physical hardware of the quantum device and find the corresponding ground state using the proper algorithm.

In Chapuis et al.[85], a D-Wave 2× Quantum Annealer has been tested for this purpose. Before discussing the results, it is convenient to divide the MIS instances into two classes. (i) problems that fit into the processor topology and (ii) problems that can't fit into the processor. Let's start from the first case, the D-Wave 2X processor has around 1000 qubits arranged in a peculiar connectivity which goes by the name of Chimera graph. This graph is quite sparsely connected, with a qubit being connected to 6 neighbors on average. This might seem like a huge limitation on the problems that we can solve using these types of devices, but there are workarounds. If the problem intended to be solved on the machine does not naturally conform to the Chimera graph structure–that is, it cannot be directly mapped onto the

machine–one may resort to the technique of minor embedding as suggested by Vinci et al.[237]. In minor embedding, one uses multiple physical qubits to represent one logical qubit of the problem instance, thus expanding the available connectivity at a cost of some physical qubits. Once a minor embedding is found that effectively represents the desired graph on the Chimera graph, the qubits intended to embody the same logical qubit are linked together through a strong ferromagnetic interaction. This ensures they act collectively as a single, larger qubit. Consequently, these aggregated logical qubits are often referred to as *chains*. Just to give a rough idea of how many resources one needs to do minor embedding, the largest fully connected graph that can be fit in the D-Wave 2X has 45 nodes.

Given this, the authors in Chapuis et al.[85] tested the performance of the D-Wave 2× on the MIS using random graphs with 45 vertices and various edge probabilities $p \in [0.3, 0.9]$. The results of the quantum machine were then compared to various classical heuristic and exact algorithms for MIS, like Simulated Annealing[238], Fast Max-Clique Finder[239], and Gurobi optimizer. For these easy instances, all approaches found the correct MIS, albeit in different runtimes, with the quantum approach having a mediocre performance and definitely no advantage. Scaling to larger graphs presents a wholly different challenge. To address this, researchers utilized m edge contractions from Chimera graphs, forming what is referred to as the $\mathcal{C}_m$ family of graphs. This approach aimed to achieve precise embedding while preserving the largest possible graph that could be embedded into the machine. Performance comparisons were made with various heuristic solvers. For large graphs (containing more than 400 nodes), the D-Wave 2× frequently emerged as the sole system capable of identifying the MIS, accomplishing this significantly faster than Simulated Annealing. It achieved a speedup just shy of 10^6 for graphs comprising 1000 nodes. Although the improvement is specific to a particular family of graphs, achieving a six-order magnitude enhancement over classical solvers is undeniably impressive. This indicates the potential utility of heuristic quantum solvers for certain graph types. However, unlike the circuit model, there's no theoretical assurance of speedup, with only a few investigations, such as refs. 240,241, exploring similar dynamics. Furthermore, it remains challenging to predict whether a specific instance will effectively benefit from quantum annealing. Indeed, a recent work[242], carrying out annealing dynamics on a system composed of neutral atom qubits to solve MIS (albeit for much smaller instances of 80 qubits maximum), failed to find speedups for random instances of unit-disk graphs.

For problems that can't directly fit into a quantum processor, i.e., case (ii), the solution involves using classical branch-and-bound algorithms[85,243,244]. These algorithms decompose the main problem into smaller, manageable subproblems, using classical branch and bound algorithms. Some of these subproblems are then processed on the quantum processor, and the results are combined to address the original problem comprehensively. This method is applied to challenges from the DIMACS benchmark, but not compared against other classical solvers. As such, while hybrid quantum-classical methods are viable and operational, it's still uncertain whether they offer advantages over entirely classical approaches. Moreover, in this case, quantifying the quantum solver's specific contribution compared to classical algorithms poses a challenge, rendering the quantum processor's utility in such hybrid setups ambiguous. Classical simulations of quantum annealing dynamics have shown potential in certain cases[245], suggesting an alternative avenue for exploration.

Despite some limitations, the application of Quantum Annealing for solving the MIS is promising, demonstrating the capability to tackle realistic instances on actual quantum devices with encouraging outcomes. Recent advancements in Quantum Annealing hardware have been remarkable, transitioning from devices with approximately 1000 qubits and about 6 couplings per qubit[85], to the latest models boasting around 5000 qubits, each with approximately 15 couplings. This enhancement significantly increases the potential to embed fully connected graphs of up to 180 nodes. The deployment of this advanced architecture may offer promising prospects, as evidenced by some findings on related challenges[246]. However, it remains to be determined whether these improvements confer a definitive advantage.

Quantum approximate optimization algorithm

Another viable approach involves employing an optimized form of Quantum Annealing on a gate-model quantum computer, known as the Quantum Approximate Optimization Algorithm (QAOA)[224]. In QAOA, just like in Quantum Annealing, the solution to the problem is encoded in the ground state of a Hamiltonian.

In QAOA, the ground state is approximated through a variational approach, utilizing a specific ansatz and optimization process. We employ the same H_I and H_P Hamiltonian defined in Eqs. (9) and (10), respectively, for solving MIS or MCP. The process begins from a uniform superposition state $|\psi\rangle_N = \frac{1}{\sqrt{2^N}} \sum_{i=0}^{2^N-1} |i\rangle$, analogous to the initial state preparation in the circuit model approach. Then, we build our variational ansatz using two sets of free parameters $\gamma = \{\gamma_1, \ldots, \gamma_p\}$ and $\beta = \{\beta_1, \ldots, \beta_p\}$ as follows:

$$|\psi(\gamma,\beta)\rangle = e^{-i\beta_p H_I} e^{-i\gamma_p H_P} \ldots e^{-i\beta_1 H_I} e^{-i\gamma_1 H_P} |\psi\rangle_N. \tag{13}$$

After this quantum circuit's operation, the system undergoes measurement in the computational basis across multiple iterations. This process is essential for calculating the expectation value of the cost function:

$$F_p(\gamma,\beta) = \langle\psi(\gamma,\beta)|H_P|\psi(\gamma,\beta)\rangle \geq E_0, \tag{14}$$

where E_0 is the ground state energy of H_P. The parameters γ and β are optimized through a classical subroutine, aiming that, after an adequate number of iterations, the state $|\psi(\gamma^*,\beta^*)\rangle$ aligns with the ground state of H_P. This state represents the solution to our MIS problem.

For simplicity, one might consider QAOA as a digital counterpart to quantum annealing. Specifically, the state described in Eq. (10) can be seen as equivalent to a Trotterized version of quantum annealing dynamics, implemented with p Trotter steps. To sum up this brief overview of QAOA, it's important to highlight that the algorithm functions as a hybrid between quantum and classical computing. The variational ansatz is executed on a quantum device, while the optimization of parameters is performed by a classical computer. This creates a synergistic loop between the classical and quantum domains to efficiently run the algorithm.

The implementation of QAOA for addressing MIS problems on a quantum processor utilizing Rydberg atoms (also known as neutral atoms) has shown promising results[247]. The groundbreaking research by Ebadi et al.[248] demonstrated the successful resolution of MIS problems for a specific family of graphs, namely unit-disk graphs, with up to 80 vertices. These graphs are particularly well-matched to the connectivity of their quantum processor, akin to the favorable scenarios in quantum annealing. Comparative analysis with Simulated Annealing revealed a scaling advantage related to the hardness parameter of the instances, indicating a nearly quadratic speedup for the quantum approach. This represents a scenario where a theoretical quantum speedup can be substantiated, contrasting with the quantum annealing method, where such proof is more elusive[112].

Another recent study by Yin et al.[96] explores the use of Rydberg atom architectures for solving problems on a distinct class of graphs known as King graphs. Addressing the significant challenge of connectivity–a major hurdle for the practical deployment of quantum algorithms–an approach proposed by Nguyen et al.[249] suggests a method for embedding problems of any connectivity with a quadratic overhead. Although this technique awaits experimental verification, it leverages the unique reconfigurability of Rydberg atom processors and their potential for qubit shuttling during computations[250]. This adaptability positions Rydberg atom-based systems as an increasingly sought-after option in the quantum computing arena.

It is worth noting here that all algorithms for MCP implemented so far in the quantum realm are heuristics; hence, we have no general guarantees in terms of complexity or scaling. Regarding the implementation of exact classical or quantum algorithms in quantum processors, there's no speedup to be expected in this domain for quantum algorithms unless P = NP[219,251].

Table 6 | Comparison of different algorithmic approaches to the MCP

Approach	Advantages	Limitations	When to use
Exact algorithms	Proven optimal solutions with strong performance on dense graphs (e.g., DIMACS benchmarks).	Exponential time complexity, scaling is limited to graphs with thousands of vertices, even for advanced solvers.	When optimality is required (e.g., verification, benchmarking) or for medium-sized dense graphs with up to ~10^3 vertices.
Heuristic algorithms	Fast, scalable to very large graphs, often sufficient for practical or real-time applications.	May find only near-optimal cliques, cannot certify optimality, approximation hardness is provably strong.	When approximate or "good enough" solutions are acceptable, especially in real-world large or sparse networks.
GNN-based algorithms	Handling complex graph structures, can generalize patterns from similar instances, adaptable to dynamic or data-driven problems.	Performance depends on training, limited theoretical guarantees, often outperformed by simple greedy algorithms on random graphs, depth limitations.	For data-rich environments or earning-based combinatorial optimization, where many similar MCP instances are available.
Quantum algorithms	Theoretical potential for speedups if proper algorithms and hardware are combined.	Current devices are noisy and small-scale; algorithms remain heuristic, no proven speedup.	For experimental or small-scale instances and quantum benchmarking, especially to test new quantum hardware.

Conclusions

In this paper, we have embarked on a comprehensive exploration of the MCP, delving into its formalism, theoretical bounds, and the plethora of algorithms–both classical, learning, and quantum–developed to tackle it. Section "Maximum clique problem" laid the groundwork by defining MCP and outlining its complexity, setting the stage for an in-depth analysis of existing solution strategies. Following this, Section "Classical algorithms for MCP" provided a thorough review of classical approaches to solving MCP, encompassing both exact algorithms and heuristic methods, and extending the discussion to Monte Carlo algorithms for the Hidden Clique Problem (HCP), aiming to cover the state-of-the-art in these domains.

In the realm of emerging technologies, Section "Learning algorithms for MCP: the graph neural networks (GNNs)" surveyed the advancements in Graph Neural Networks (GNNs) and their application to solving MCP, demonstrating the potential of leveraging machine learning techniques in combinatorial optimization.

The quantum computing frontier, covered in Section "Quantum algorithms for MCP", presents the latest quantum algorithms for MCP, signifying a leap towards harnessing quantum mechanics for computational advantage in solving notoriously difficult problems.

As quantum hardware evolves and becomes more capable of handling complex computations, the benchmarks outlined in this paper will serve as a critical resource for the community. They not only facilitate the rigorous testing of new algorithms but also encourage the development of more sophisticated and efficient solutions. By laying out the current landscape and future directions, this work aims to spur further research (both in quantum and classical) and collaboration in tackling MCP and similar combinatorial optimization problems.

As we have described above, the types of solvers covered in this review are very different and adapt well to different scenarios. Some of them are exact, some heuristic, some work well on large instances, some on small ones. This makes a cross-benchmark between all of these techniques meaningless for all practical purposes. In general, the choice of approach for any specific problem will depend on a variety of factors such as the scale of the problem, the structure of the graph (i.e., dense or sparse graphs), the available computational resources and the balance sought between provable accuracy and computational feasibility. We summarized the main take-aways for each algorithmic family in Table 6.

In conclusion, this review contributes to the foundational understanding of MCP and offers insights into the multifaceted approaches developed to address it. As the field of quantum computing continues to mature, it holds the promise of unlocking new possibilities in solving combinatorial optimization problems, potentially surpassing the limitations of classical computing. The journey from theoretical formalism to practical application underscores the dynamic and evolving nature of computational science, beckoning a future where these complex problems can be addressed more efficiently.

Data availability

The benchmark datasets presented in this paper and the code used to generate them are publicly available at https://github.com/zbogdan/evil2 and https://github.com/zbogdan/qbenchmarks.

Received: 9 June 2025; Accepted: 19 March 2026;
Published online: 16 July 2026

References

1. Luce, R. D. & Perry, A. D. A method of matrix analysis of group structure. *Psychometrika* **14**, 95–116 (1949).
2. Harary, F. & Ross, I. C. A procedure for clique detection using the group matrix. *Sociometry* **20**, 205–215 (1957).
3. Okamoto, Y. Finding a maximum common subgraph from molecular structural formulas through the maximum clique approach combined with the ising model. *ACS Omega* **5**, 13064–13068 (2020).
4. Meng, X.-Y., Zhang, H.-X., Mezei, M. & Cui, M. Molecular docking: a powerful approach for structure-based drug discovery. *Curr. Comput. Aided Drug Des.* **7**, 146–157 (2011).
5. Das, N. R., Chaudhury, K. N. & Pal, D. Improved NMR-data-compliant protein structure modeling captures context-dependent variations and expands the scope of functional inference. *Proteins* **91**, 412–435 (2023).
6. Strickland, D. M., Barnes, E. & Sokol, J. S. Optimal protein structure alignment using maximum cliques. *Oper. Res.* **53**, 389–402 (2005).
7. Konc, J. & Janežič, D. ProBiS algorithm for detection of structurally similar protein binding sites by local structural alignment. *Bioinformatics* **26**, 1160–1168 (2010).
8. Fine, J., Konc, J., Samudrala, R. & Chopra, G. CANDOCK: Chemical atomic network-based hierarchical flexible docking algorithm using generalized statistical potentials. *J. Chem. Inf. Model.* **60**, 1509–1527 (2020).
9. Rozman, K. et al. Enhanced molecular docking: novel algorithm for identifying highest weight *k*-cliques in weighted general and protein-ligand graphs. *J. Mol. Struct.* **1304**, 137639 (2024).
10. Makarenkov, V., Barseghyan, G. S. & Tahiri, N. Inferring multiple consensus trees and supertrees using clustering: A review. In *Data Analysis and Optimization: In Honor of Boris Mirkin's 80th Birthday* (eds Goldengorin, B. & Kuznetsov, S.) 191–213 (Springer Nature Switzerland, Cham, 2023).
11. Heal, M., Dashtipour, K. & Gogate, M. The P vs. NP problem and attempts to settle it via perfect graphs state-of-the-art approach. In *Advances in Information and Communication* (ed. Arai, K.) 328–340 (Springer Nature Switzerland, Cham, 2023).
12. Corrádi, K. & Szabó, S. A combinatorial approach for Keller's conjecture. *Period. Math. Hung.* **21**, 95–100 (1990).
13. Lagarias, J. C. & Shor, P. W. Keller's cube-tiling conjecture is false in high dimensions. *Bull. Am. Math. Soc.* **27**, 279–283 (1992).

14. Brakensiek, J., Heule, M., Mackey, J. & Narváez, D. The resolution of Keller's conjecture. In *Automated Reasoning* (eds Peltier, N. & Sofronie-Stokkermans, V.) 48–65 (Springer International Publishing, Cham, 2020).
15. Ashlock, D., Houghten, S. K., Brown, J. A. & Orth, J. On the synthesis of DNA error correcting codes. *Biosystems* **110**, 1–8 (2012).
16. Battail, G. Error-correcting codes and information in biology. *Biosystems* **184**, 103987 (2019).
17. Etzion, T. & Östergård, P. Greedy and heuristic algorithms for codes and colorings. *IEEE Trans. Inf. Theory* **44**, 382–388 (1998).
18. Boginski, V., Butenko, S. & Pardalos, P. M. Statistical analysis of financial networks. *Comput. Stat. Data Anal.* **48**, 431–443 (2005).
19. Ganesan, A. Structured hypergraphs in cellular mobile communication systems. In *Proc. 24th International Conference on Distributed Computing and Networking, ICDCN '23* 188–196 (Association for Computing Machinery, New York, NY, USA, 2023).
20. Dai, Y. et al. Joint mode selection and resource allocation for D2D-enabled NOMA cellular networks. *IEEE Trans. Veh. Technol.* **68**, 6721–6733 (2019).
21. Zhu, F., Zhang, C., Zheng, Z. & Farouk, A. Practical network coding technologies and softwarization in wireless networks. *IEEE Internet Things J.* **8**, 5211–5218 (2021).
22. Douik, A., Dahrouj, H., Al-Naffouri, T. Y. & Alouini, M.-S. A tutorial on clique problems in communications and signal processing. *Proc. IEEE* **108**, 583–608 (2020).
23. Rajsbaum, S. A distributed computing perspective of unconditionally secure information transmission in Russian cards problems. *Theor. Comput. Sci.* **952**, 113761 (2023).
24. Zhong, H., Pajouh, F. M. & Prokopyev, O. A. On designing networks resilient to clique blockers. *Eur. J. Oper. Res.* **307**, 20–32 (2023).
25. Lu, Y., Miao, Z., Sahraeian, P. & Balasundaram, B. On atomic cliques in temporal graphs. *Optim. Lett.* **17**, 1–16 (2023).
26. Grandi, U., Kanesh, L., Lisowski, G., Ramanujan, M. & Turrini, P. A complexity-theoretic analysis of majority illusion in social networks. *J. Artif. Int. Res.* **83**, 26 (2025).
27. Szabó, S. & Zaválnij, B. Clique search in graphs of special class and job shop scheduling. *Mathematics* **10**, 697 (2022).
28. Zhong, H., Mahdavi Pajouh, F. & Prokopyev, O. On designing networks resilient to clique blockers. *Eur. J. Oper. Res.* **307**, 20–32 (2023).
29. Nasirian, F., Mahdavi Pajouh, F. & Balasundaram, B. Detecting a most closeness-central clique in complex networks. *Eur. J. Oper. Res.* **283**, 461–475 (2020).
30. Rysz, M., Mahdavi Pajouh, F. & Pasiliao, E. L. Finding clique clusters with the highest betweenness centrality. *Eur. J. Oper. Res.* **271**, 155–164 (2018).
31. Angelini, M. C. & Ricci-Tersenghi, F. Limits and performances of algorithms based on simulated annealing in solving sparse hard inference problems. *Phys. Rev. X* **13**, 021011 (2023).
32. Angelini, M. C., Cavaliere, A. G., Marino, R. & Ricci-Tersenghi, F. Stochastic gradient descent-like relaxation is equivalent to Metropolis dynamics in discrete optimization and inference problems. *Sci. Rep.* **14**, 11638 (2024).
33. Schaeffer, S. E. Graph clustering. *Comput. Sci. Rev.* **1**, 27–64 (2007).
34. Guo, L. & Dai, Q. Graph clustering via variational graph embedding. *Pattern Recognit.* **122**, 108334 (2022).
35. Bóta, A., Krész, M. & Zaválnij, B. Adaptations of the *k*-means algorithm to community detection in parallel environments. In *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* 299–302 (2015).
36. Marino, R., Parisi, G. & Ricci-Tersenghi, F. The backtracking survey propagation algorithm for solving random *k*-SAT problems. *Nat. Commun.* **7**, 12996 (2016).
37. Marino, R. & Kirkpatrick, S. Large independent sets on random d-regular graphs with fixed degree d. *Computation* **11**, 206 (2023).
38. Szabó, S. & Zaválnij, B. Graph coloring via clique search with symmetry breaking. *Symmetry* **14**, 1574 (2022).
39. Jovanovic, R., Sanfilippo, A. P. & Voß, S. Fixed set search applied to the clique partitioning problem. *Eur. J. Oper. Res.* **309**, 65–81 (2023).
40. Zhou, Q., Benlic, U. & Wu, Q. An opposition-based memetic algorithm for the maximum quasi-clique problem. *Eur. J. Oper. Res.* **286**, 63–83 (2020).
41. Pinto, B. Q., Ribeiro, C. C., Rosseti, I. & Plastino, A. A biased random-key genetic algorithm for the maximum quasi-clique problem. *Eur. J. Oper. Res.* **271**, 849–865 (2018).
42. Zhou, Y., Hao, J.-K. & Goëffon, A. Push: A generalized operator for the maximum vertex weight clique problem. *Eur. J. Oper. Res.* **257**, 41–54 (2017).
43. Pardalos, P. M. & Xue, J. The maximum clique problem. *J. Glob. Optim.* **4**, 301–328 (1994).
The first comparative review on the maximum clique problem.
44. Bomze, I. M., Budinich, M., Pardalos, P. M. & Pelillo, M. The maximum clique problem. In *Handbook of Combinatorial Optimization: Supplement* Volume A 1-74 (Springer, 1999).
A summative review on the maximum clique problem, including exact and heuristic approach and applications.
45. Wu, Q. & Hao, J.-K. A review on algorithms for maximum clique problems. *Eur. J. Oper. Res.* **242**, 693–709 (2015).
The latest review previously up do date focusing on exact and heuristic approaches.
46. Feldmann, A. E., Karthik, C. S., Lee, E. & Manurangsi, P. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms* **13**, 146 (2020).
47. Karp, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department* (eds Miller, R. E. et al.) 85–103 (Springer US, Boston, MA, 1972).
48. Impagliazzo, R. & Paturi, R. On the complexity of *k*-SAT. *J. Comput. Syst. Sci.* **62**, 367–375 (2001).
49. McCreesh, C., Pettersson, W. & Prosser, P. Understanding the empirical hardness of random optimisation problems. In *Principles and Practice of Constraint Programming* (eds Schiex, T. & de Givry, S.) 333–349 (Springer International Publishing, Cham, 2019).
50. Downey, R. G. & Fellow, M. R. *Parameterized Complexity*. Monographs in Computer Science 1st edn (Springer, 1999).
51. Downey, R. G. & Fellow, M. R. *Fundamentals of Parameterized Complexity.* Texts in Computer Science 1st edn (Springer, 2013).
52. Håstad, J. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta. Math.* **182**, 105–142 (1999).
53. Feige, U. Approximating maximum clique by removing subgraphs. *SIAM J. Discret. Math.* **18**, 219–225 (2004).
54. Engebretsen, L. & Holmerin, J. Towards optimal lower bounds for clique and chromatic number. *Theor. Comput. Sci.* **299**, 537–584 (2003).
55. Zuckerman, D. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proc. Thirty-Eighth Annual ACM Symposium on Theory of Computing, STOC '06* 681–690 (Association for Computing Machinery, New York, NY, USA, 2006).
56. Chitnis, R., Hajiaghayi, M. & Kortsarz, G. Fixed-parameter and approximation algorithms: a new look. In *Parameterized and Exact Computation* (eds Gutin, G. & Szeider, S.) 110–122 (Springer International Publishing, Cham, 2013).
57. Chalermsook, P. et al. From gap-exponential time hypothesis to fixed parameter tractable inapproximability: Clique, dominating set, and more. *SIAM J. Comput.* **49**, 772–810 (2020).
58. Jerrum, M. Large cliques elude the Metropolis process. *Random Struct. Algor.* **3**, 347–359 (1992).

59. Alon, N., Krivelevich, M. & Sudakov, B. Finding a large hidden clique in a random graph. *Random Struct. Algor.* **13**, 457–466 (1998).
60. Dekel, Y., Gurel-Gurevich, O. & Peres, Y. Finding hidden cliques in linear time with high probability. *Comb. Probab. Comput.* **23**, 29–49 (2014).
61. Deshpande, Y. & Montanari, A. Finding hidden cliques of size $\sqrt{N/e}$ in nearly linear time. *Found. Comput. Math.* **15**, 1069–1128 (2015).
62. Hopkins, S. B. et al. The power of sum-of-squares for detecting hidden structures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)* 720–731 (IEEE Computer Society, Los Alamitos, CA, USA, 2017).
63. Lasserre, J. B. Global optimization with polynomials and the problem of moments. *SIAM J. Optim.* **11**, 796–817 (2001).
64. Meka, R., Potechin, A. & Wigderson, A. Sum-of-squares lower bounds for planted clique. In *Proc. Forty-Seventh Annual ACM Symposium on Theory of Computing, STOC '15* 87–96 (Association for Computing Machinery, New York, NY, USA, 2015).
65. Deshpande, Y. & Montanari, A. Improved sum-of-squares lower bounds for hidden clique and hidden submatrix problems. In *Proc. 28th Conference on Learning Theory Proceedings of Machine Learning Research*. Vol. 40 (eds Grünwald, P. et al.) 523–562 (PMLR, Paris, France, 2015).
66. Pang, S. SOS lower bound for exact planted clique. In *Proc. 36th Computational Complexity Conference, CCC '21* (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, DEU, 2021).
67. Jones, C., Potechin, A., Rajendran, G., Tulsiani, M. & Xu, J. Sum-of-squares lower bounds for sparse independent set. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)* 406–416 (IEEE, 2022).
68. Jones, C., Potechin, A., Rajendran, G. & Xu, J. Sum-of-squares lower bounds for densest k-subgraph. In *Proc. 55th Annual ACM Symposium on Theory of Computing, STOC 2023* 84–95 (Association for Computing Machinery, New York, NY, USA, 2023).
69. De Rezende, S. F., Potechin, A. & Risse, K. Clique is hard on average for unary Sherali-Adams. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)* 12–25 (2023).
70. Hosseinian, S. et al. The maximum edge weight clique problem: Formulations and solution approaches. In *Optimization Methods and Applications: In Honor of Ivan V. Sergienko's 80th Birthday* (eds Butenko, S. et al.) 217–237 (Springer International Publishing, 2017).
71. Nemhauser, G. L. & Trotter, L. E. Jr. Vertex packings: structural properties and algorithms. *Math. Program.* **8**, 232–248 (1975).
72. Nemhauser, G. L. & Trotter, L. E. Jr. Properties of vertex packing and independence system polyhedra. *Math. Program.* **6**, 48–61 (1974).
73. Moon, J. & Moser, L. On cliques in graphs. *Israel J. Math.* **3**, 22–28 (1965).
74. Gouveia, L. & Martins, P. Solving the maximum edge-weight clique problem in sparse graphs with compact formulations. *EURO J. Comput. Optim.* **3**, 1–30 (2015).
75. Hosseinian, S., Fontes, D. B. & Butenko, S. A nonconvex quadratic optimization approach to the maximum edge weight clique problem. *J. Glob. Optim.* **72**, 219–240 (2018).
76. Shor, N. Z. Dual quadratic estimates in polynomial and boolean programming. *Ann. Oper. Res.* **25**, 163–168 (1990).
77. Motzkin, T. S. & Straus, E. G. Maxima for graphs and a new proof of a theorem of Turán. *Can. J. Math.* **17**, 533–540 (1965).
78. Martins, P. Extended and discretized formulations for the maximum clique problem. *Comput. Oper. Res.* **37**, 1348–1358 (2010).
79. Bollobas, B. & Erdős, P. Cliques in random graphs. *Math. Proc. Camb. Philos. Soc.* **80**, 419–427 (1976).
80. Matula, D. W. The largest clique size in a random graph. *Tech. Rep.* CS 7608. https://s2.smu.edu/matula/Tech-Report76.pdf (Department of Computer Science, Southern Methodist University, 1976).
81. Matula, D. W. On the complete subgraphs of a random graph. In *Proc. of the Second Chapel Hill Conference on Combinatorial Mathematics and Its Applications* 356–369 (University of North Carolina, Chapel Hill, 1970).
82. Matula, D. W. Employee party problem. *Not. Am. Math. Soc.* **19**, A382 (1972).
83. Belachew, M. T. & Gillis, N. Solving the maximum clique problem with symmetric rank-one non-negative matrix approximation. *J. Optim. Theory Appl.* **173**, 279–296 (2017).
84. Lucas, A. Ising formulations of many NP problems. *Front. Phys.* **2**, 5 (2014).
85. Chapuis, G., Djidjev, H., Hahn, G. & Rizk, G. Finding maximum cliques on a quantum annealer. In *Proc. Computing Frontiers Conference, CF'17* 63–70 (Association for Computing Machinery, New York, NY, USA, 2017).
An investigation into the performance of Quantum Annealers in solving various MCP instances.
86. Johnson, D. J. & Trick, M. A. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993* (American Mathematical Society, USA, 1996).
87. Jin, Y. & Hao, J.-K. General swap-based multiple neighborhood tabu search for the maximum independent set problem. *Eng. Appl. Artif. Intell.* **37**, 20–33 (2015).
88. Best, M., Brouwer, A., MacWilliams, F., Odlyzko, A. & Sloane, N. Bounds for binary codes of length less than 25. *IEEE Trans. Inf. Theory* **24**, 81–93 (1978).
89. Brouwer, A., Shearer, J., Sloane, N. & Smith, W. A new table of constant weight codes. *IEEE Trans. Inf. Theory* **36**, 1334–1380 (1990).
90. Li, C.-M., Jiang, H. & Manyà, F. On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem. *Comput. Oper. Res.* **84**, 1–15 (2017).
91. Xu, K., Boussemart, F., Hemery, F. & Lecoutre, C. Random constraint satisfaction: easy generation of hard (satisfiable) instances. *Artif. Intell.* **171**, 514–534 (2007).
92. Xu, K., Boussemart, F., Hemery, F. & Lecoutre, C. A simple model to generate hard satisfiable instances. In *Proc. 19th International Joint Conference on Artificial Intelligence, IJCAI'05* 337–342 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005).
93. Jin, Y., Hamiez, J.-P. & Hao, J.-K. Algorithms for the minimum sum coloring problem: a review. *Artif. Intell. Rev.* **47**, 367–394 (2017).
94. Pan, S. et al. An improved master-apprentice evolutionary algorithm for minimum independent dominating set problem. *Front. Comput. Sci.* **17**, 174326 (2023).
95. Xiao, M., Huang, S., Zhou, Y. & Ding, B. Efficient reductions and a fast algorithm of maximum weighted independent set. In *Proc. Web Conference 2021* 3930–3940 (2021).
96. Yin, X.-F. et al. Solving independent set problems with photonic quantum circuits. *Proc. Natl. Acad. Sci. USA* **120**, e2212323120 (2023).
97. Cai, S., Lin, J. & Luo, C. Finding a small vertex cover in massive sparse graphs: Construct, local search, and preprocess. *J. Artif. Intell. Res.* **59**, 463–494 (2017).
98. Szabó, S. & Zavalnij, B. Benchmark problems for exhaustive exact maximum clique search algorithms. *Informatica* **43**, 177–186 (2019).
99. Sloane, N. J. A. Challenge problems: independent sets in graphs. https://oeis.org/A265032/a265032.html (2000).
100. Nakasho, K., Hagiwara, M., Anderson, A. & Nation, J. B. The tight upper bound for the size of single deletion error correcting codes of length 11. In *2024 International Symposium on Information Theory and Its Applications (ISITA)* 389–394 (IEEE, 2024).
101. Dzulfikar, M. A., Fichte, J. K. & Hecher, M. The PACE2019 parameterized algorithms and computational experiments challenge: the fourth iteration. In *14th International Symposium on Parameterized and Exact Computation (IPEC 2019). Leibniz International Proceedings in Informatics* (LIPIcs). Vol. 148 (eds

Jansen, B. M. P. & Telle, J. A.) 25:1–25:23 (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2019).
102. Fichte, J. & Hecher, M. PACE 2019. https://github.com/daajoe/pace2019_vc_instances/blob/master/0_instance_selection/PACE2019_benchmark_selection_vc_mapping_selected_original.csv (2019).
103. San Segundo, P., Furini, F., Álvarez, D. & Pardalos, P. M. CliSAT: A new exact algorithm for hard maximum clique problems. *Eur. J. Oper. Res.* **307**, 1008–1025 (2023).
The paper on the latest state-of-the-art-exact MCP solver.
104. Aragón Artacho, F. J. & Campoy, R. Solving graph coloring problems with the Douglas-Rachford algorithm. *Set-Valued Var. Anal.* **26**, 277–304 (2018).
105. Prates, M. & Lamb, L. Problem solving at the edge of chaos: Entropy, puzzles and the sudoku freezing transition. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)* 686–693 (IEEE, 2018).
106. Kriege, N. M., Neumann, M., Morris, C., Kersting, K. & Mutzel, P. A unifying view of explicit and implicit feature maps of graph kernels. *Data Mining Knowl. Discov.* **33**, 1505–1547 (2019).
107. Yanardag, P. & Vishwanathan, S. Deep graph kernels. In *Proc. 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15* 1365–1374 (Association for Computing Machinery, New York, NY, USA, 2015).
108. Leskovec, J. & Sosič, R. Snap: A general-purpose network analysis and graph-mining library. *ACM Trans. Intell. Syst. Technol.* **8**, 1–20 (2016).
109. Traud, A. L., Mucha, P. J. & Porter, M. A. Social structure of Facebook networks. *Phys. A Stat. Mech. Appl.* **391**, 4165–4180 (2012).
110. Liu, J.-G., Gao, X., Cain, M., Lukin, M. D. & Wang, S.-T. Computing solution space properties of combinatorial optimization problems via generic tensor networks. *SIAM J. Sci. Comput.* **45**, A1239–A1270 (2023).
111. Bojić, A. Quantum algorithm for finding a maximum clique in an undirected graph. *J. Inf. Organ. Sci.* **36**, 91–98 (2012).
112. Cain, M. et al. Quantum speedup for combinatorial optimization with flat energy landscapes. Preprint at https://arXiv.org/abs/2306.13123 (2023).
113. Lawler, E. L. & Wood, D. E. Branch-and-bound methods: a survey. *Oper. Res.* **14**, 699–719 (1966).
114. Bron, C. & Kerbosch, J. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM* **16**, 575–577 (1973).
115. Carraghan, R. & Pardalos, P. M. An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* **9**, 375–382 (1990).
116. Balas, E. & Yu, C. S. Finding a maximum clique in an arbitrary graph. *SIAM J. Comput.* **15**, 1054–1068 (1986).
117. Konc, J. & Janežič, D. An improved branch and bound algorithm for the maximum clique problem. *MATCH Commun. Math. Comput. Chem.* **58**, 569–590 (2007).
118. Tomita, E. Efficient algorithms for finding maximum and maximal cliques and their applications. In *WALCOM: Algorithms and Computation* (eds Poon, S.-H. et al.) 3–15 (Springer International Publishing, Cham, 2017).
119. Östergård, P. R. A fast algorithm for the maximum clique problem. *Discret. Appl. Math.* **120**, 197–207 (2002).
120. Pardalos, P. M., Rappe, J. & Resende, M. G. C. An exact parallel algorithm for the maximum clique problem. In *High Performance Algorithms and Software in Nonlinear Optimization* (eds De Leone, R. et al.) 279–300 (Springer US, Boston, MA, 1998).
121. Golomb, S. W. & Baumert, L. D. Backtrack programming. *J. ACM* **12**, 516–524 (1965).
122. Margenov, S. et al. Applications for ultrascale systems. In *Ultrascale Computing Systems, Computing* 189–244 (Institution of Engineering and Technology, 2019).
123. Ebenegger, C., Hammer, P. & de Werra, D. Pseudo-boolean functions and stability of graphs. In *Algebraic and Combinatorial Methods in Operations Research* Vol. 95 (eds Burkard, R. et al.) 83–97 (North-Holland, 1984).
124. Alexe, G., Hammer, P. L., Lozin, V. V. & de Werra, D. Struction revisited. *Discret. Appl. Math.* **132**, 27–46 (2003).
125. Bodlaender, H. L., Downey, R. G., Fellows, M. R. & Hermelin, D. On problems without polynomial kernels. *J. Comput. Syst. Sci.* **75**, 423–434 (2009).
126. Hespe, D., Schulz, C. & Strash, D. Scalable kernelization for maximum independent sets. *ACM J. Exp. Algor.* **24**, 1.16 (2019).
127. Strash, D. On the power of simple reductions for the maximum independent set problem. In *Computing and Combinatorics* (eds Dinh, T. N. & Thai, M. T.) 345–356 (Springer International Publishing, Cham, 2016).
128. Niedermeier, R. *Invitation to Fixed-Parameter Algorithms* (Oxford University Press, 2006).
129. Akiba, T. & Iwata, Y. Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover. *Theor. Comput. Sci.* **609**, 211–225 (2016).
130. Cygan, M. et al. *Parameterized Algorithms* (Springer Cham, 2015).
131. Chang, L. Efficient maximum clique computation over large sparse graphs. In *Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19* 529–538 (Association for Computing Machinery, New York, NY, USA, 2019).
132. Chang, L. Efficient maximum clique computation and enumeration over large sparse graphs. *VLDB J.* **29**, 999–1022 (2020).
133. Rossi, R. A., Gleich, D. F. & Gebremedhin, A. H. Parallel maximum clique algorithms with applications to network analysis. *SIAM J. Sci. Comput.* **37**, C589–C616 (2015).
134. Gellner, A., Lamm, S., Schulz, C., Strash, D. & Zavalnij, B. Boosting data reduction for the maximum weight independent set problem using increasing transformations. In *Proc. Symposium on Algorithm Engineering and Experiments (ALENEX)* 128–142 (2021).
The latest state-of-the-art paper on preconditioning and exact solving of weighted MCP, especially focusing on preconditioning and reductions.
135. San Segundo, P., Nikolaev, A. & Batsyn, M. Infra-chromatic bound for exact maximum clique search. *Comput. Oper. Res.* **64**, 293–303 (2015).
136. San Segundo, P., Nikolaev, A., Batsyn, M. & Pardalos, P. M. Improved infra-chromatic bound for exact maximum clique search. *Informatica* **27**, 463–487 (2016).
137. San Segundo, P., Artieda, J., Leon, R. & Tapia, C. An enhanced infra-chromatic bound for the maximum clique problem. In *Machine Learning, Optimization, and Big Data: Second International Workshop, MOD 2016, Volterra, Italy, August 26-29, 2016, Revised Selected Papers 2* 306–316 (Springer, 2016).
138. Segundo, P. S., Lopez, A., Batsyn, M., Nikolaev, A. & Pardalos, P. M. Improved initial vertex ordering for exact maximum clique search. *Appl. Intell.* **45**, 868–880 (2016).
139. Komosko, L., Batsyn, M., Segundo, P. S. & Pardalos, P. M. A fast greedy sequential heuristic for the vertex colouring problem based on bitwise operations. *J. Comb. Optim.* **31**, 1665–1677 (2016).
140. Li, C.-M. & Quan, Z. An efficient branch-and-bound algorithm based on MaxSAT for the maximum clique problem. *Proc. AAAI Conf. Artif. Intell.* **24**, 128–133 (2010).
141. Jiang, H., Li, C.-M. & Manyà, F. Combining efficient preprocessing and incremental MaxSAT reasoning for maxclique in large graphs. In *Proc. Twenty-Second European Conference on Artificial Intelligence, ECAI'16* 939–947 (IOS Press, NLD, 2016).
142. Li, C.-M., Fang, Z., Jiang, H. & Xu, K. Incremental upper bound for the maximum clique problem. *INFORMS J. Comput.* **30**, 137–153 (2018).
143. Szabó, S. & Zavalnij, B. A different approach to maximum clique search. In *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* 310–316 (2018).

144. Ouyang, M. How good are branching rules in DPLL? *Discret. Appl. Math.* **89**, 281–286 (1998).
145. Szabó, S. & Zavalnij, B. Decomposing clique search problems into smaller instances based on node and edge colorings. *Discret. Appl. Math.* **242**, 118–129 (2018).
146. Li, C.-M., Liu, Y., Jiang, H., Manyà, F. & Li, Y. A new upper bound for the maximum weight clique problem. *Eur. J. Oper. Res.* **270**, 66–77 (2018).
147. Depolli, M., Konc, J., Rozman, K., Trobec, R. & Janežič, D. Exact parallel maximum clique algorithm for general and protein graphs. *J. Chem. Inf. Model.* **53**, 2217–2228 (2013).
148. Zavalnij, B. Speeding up parallel combinatorial optimization algorithms with Las Vegas method. In *Large-Scale Scientific Computing* (eds Lirkov, I. et al.) 258–266 (Springer International Publishing, Cham, 2015).
149. Hrga, T. Parallel branch and bound algorithm for the stable set problem. https://ug.zib.de/workshop2019/ (2019). Talk on the Workshop on Parallel Algorithms in Tree Search and Mathematical Optimization.
150. Jiang, H., Li, C.-M. & Manya, F. An exact algorithm for the maximum weight clique problem in large graphs. *Proc. AAAI Conf. Artif. Intell.* **31**, https://doi.org/10.1609/aaai.v31i1.10648 (2017).
151. Jiang, H., Li, C.-M., Liu, Y. & Manyà, F. A two-stage MaxSAT reasoning approach for the maximum weight clique problem. *Proc. AAAI Conf. Artif. Intell.* **32**, https://doi.org/10.1609/aaai.v32i1.11527 (2018).
152. San Segundo, P., Furini, F. & Artieda, J. A new branch-and-bound algorithm for the maximum weighted clique problem. *Comput. Oper. Res.* **110**, 18–33 (2019).
153. San Segundo, P., Coniglio, S., Furini, F. & Ljubić, I. A new branch-and-bound algorithm for the maximum edge-weighted clique problem. *Eur. J. Oper. Res.* **278**, 76–90 (2019).
154. Shimizu, S., Yamaguchi, K. & Masuda, S. A branch-and-bound based exact algorithm for the maximum edge-weight clique problem. In *Computational Science/Intelligence & Applied Informatics* (ed. Lee, R.) 27–47 (Springer International Publishing, Cham, 2019).
155. Furini, F., Ljubić, I., San Segundo, P. & Zhao, Y. A branch-and-cut algorithm for the edge interdiction clique problem. *Eur. J. Oper. Res.* **294**, 54–69 (2021).
156. Coniglio, S., Furini, F. & San Segundo, P. A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts. *Eur. J. Oper. Res.* **289**, 435–455 (2021).
157. Hespe, D., Lamm, S., Schulz, C. & Strash, D. WeGotYouCovered: The winning solver from the PACE 2019 challenge, vertex cover track. In *Proc. SIAM Workshop on Combinatorial Scientific Computing (CSC)* 1–11 (2020).
158. Walteros, J. L. & Buchanan, A. Why is maximum clique often easy in practice? *Oper. Res.* **68**, 1866–1895 (2020).
159. Kučera, L. Expected complexity of graph partitioning problems. *Discret. Appl. Math.* **57**, 193–212 (1995).
160. Feige, U. & Ron, D. Finding hidden cliques in linear time. In *DMTCS Proceedings, vol. DMTCS Proceedings vol. AM, 21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10)* (eds Drmota, et al.) 189–204 (Discrete Mathematics and Theoretical Computer Science, Nancy, France, 2010).
161. Li, Q. & Sompolinsky, H. Statistical mechanics of deep linear neural networks: The backpropagating kernel renormalization. *Phys. Rev. X* **11**, 031059 (2021).
162. Zhao, C.-Y. & Fu, Y.-R. Belief propagation guided decimation algorithms for random constraint satisfaction problems with growing domains. *J. Stat. Mech. Theory Exp.* **2021**, 033408 (2021).
163. Crotti, S. & Braunstein, A. Matrix product belief propagation for reweighted stochastic dynamics over graphs. *Proc. Natl. Acad. Sci. USA* **120**, e2307935120 (2023).
164. Pan, F., Zhou, P., Zhou, H.-J. & Zhang, P. Solving statistical mechanics on sparse graphs with feedback-set variational autoregressive networks. *Phys. Rev. E* **103**, 012103 (2021).
165. Caracciolo, S. et al. Criticality and conformality in the random dimer model. *Phys. Rev. E* **103**, 042127 (2021).
166. Zhao, Y., Qiu, J., Xie, M. & Huang, H. Equivalence between belief propagation instability and transition to replica symmetry breaking in perceptron learning systems. *Phys. Rev. Res.* **4**, 023023 (2022).
167. Old, J. & Rispler, M. Generalized belief propagation algorithms for decoding of surface codes. *Quantum* **7**, 1037 (2023).
168. Mann, P. & Dobson, S. Belief propagation on networks with cliques and chordless cycles. *Phys. Rev. E* **107**, 054303 (2023).
169. Zhao, J.-H. A local algorithm and its percolation analysis of bipartite *z*-matching problem. *J. Stat. Mech. Theory Exp.* **2023**, 053401 (2023).
170. Xu, W., Zhang, Z. & Zhou, G. Generating hard satisfiable instances by planting into random constraint satisfaction problem model with growing constraint scope length. *Phys. A Stat. Mech. Appl.* **609**, 128367 (2023).
171. Leisenberger, H., Pernkopf, F. & Knoll, C. Fixing the Bethe approximation: How structural modifications in a graph improve belief propagation. In *Proc. Thirty-Eighth Conference on Uncertainty in Artificial Intelligence. Proceedings of Machine Learning Research*. Vol. 180 (eds Cussens, J. & Zhang, K.) 1085–1095 (PMLR, 2022).
172. Nagano, Y. & Hukushima, K. Phase transition in compressed sensing with horseshoe prior. *Phys. Rev. E* **107**, 034126 (2023).
173. Aurell, E., Barbier, J., Decelle, A. & Mulet, R. The mighty force: Statistical inference and high-dimensional statistics. In *Spin Glass Theory and Far Beyond* Ch. 22, 423–451 (World Scientific, 2023).
174. Montanari, A., Ricci-Tersenghi, F. & Semerjian, G. Solving constraint satisfaction problems through belief propagation-guided decimation. In *Proc. 45th Annual Allerton Conference on Communication, Control and Computing*. 352–359 (University of Illinois, 2007).
175. Felzenszwalb, P. & Huttenlocher, D. Efficient belief propagation for early vision. In *Proc. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR* Vol. 1, I (2004).
176. Yedidia, J. S., Freeman, W. & Weiss, Y. Generalized belief propagation. In *Advances in Neural Information Processing Systems* Vol. 13 (eds Leen, T. et al.) (MIT Press, 2000).
177. Mezard, M. & Montanari, A. *Information, Physics, and Computation* (Oxford University Press, Inc., USA, 2009).
178. Frey, B. J. & MacKay, D. J. C. A revolution: belief propagation in graphs with cycles. In *Proc. 10th International Conference on Neural Information Processing Systems, NIPS'97* 479–485 (MIT Press, Cambridge, MA, USA, 1997).
179. Mooij, J. M. & Kappen, H. J. Sufficient conditions for convergence of loopy belief propagation. In *Proc. Twenty-First Conference on Uncertainty in Artificial Intelligence, UAI'05* 396–403 (AUAI Press, Arlington, VA, USA, 2005).
180. Montanari, A. Finding one community in a sparse graph. *J. Stat. Phys.* **161**, 273–299 (2015).
181. Angelini, M. C. Parallel tempering for the planted clique problem. *J. Stat. Mech. Theory Exp.* **2018**, 073404 (2018).
182. Hansmann, U. H. Parallel tempering algorithm for conformational studies of biological molecules. *Chem. Phys. Lett.* **281**, 140–150 (1997).
183. Gamarnik, D. & Zadik, I. The landscape of the planted clique problem: Dense subgraphs and the overlap gap property. *Ann. Appl. Probab.* **34**, 3375–3434 (2024).
184. Gamarnik, D. The overlap gap property: A topological barrier to optimizing over random structures. *Proc. Natl. Acad. Sci. USA* **118**, e2108492118 (2021).
185. Angelini, M. C., Fachin, P. & de Feo, S. Mismatching as a tool to enhance algorithmic performances of Monte Carlo methods for the planted clique model. *J. Stat. Mech. Theory Exp.* **2021**, 113406 (2021).

186. Chen, Z., Mossel, E. & Zadik, I. Almost-Linear Planted Cliques Elude the Metropolis Process. *Random Struct. Alg*. **66**, e21274 (2025).
187. Marino, R. & Kirkpatrick, S. Hard optimization problems have soft edges. *Sci. Rep.* **13**, 3671 (2023).
188. Gheissari, R., Jagannath, A. & Xu, Y. Finding planted cliques using gradient descent. *SIAM J. Math. Data Sci.* **7**, 643–669 (2025).
189. Lauri, J., Dutta, S., Grassia, M. & Ajwani, D. Learning fine-grained search space pruning and heuristics for combinatorial optimization. *J. Heur.* **29**, 1–35 (2023).
190. Marino, R. & Macris, N. Solving non-linear Kolmogorov equations in large dimensions by using deep learning: a numerical comparison of discretization schemes. *J. Sci. Comput.* **94**, 8 (2023).
191. Marino, R. Learning from survey propagation: a neural network for MAX-E-3-SAT. *Mach. Learn. Sci. Technol.* **2**, 035032 (2021).
192. Chicchi, L., Giambagli, L., Buffoni, L., Marino, R. & Fanelli, D. Complex recurrent spectral network. *Chaos Solitons Fractals* **184**, 114998 (2024).
193. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).
First computational method that can regularly predict protein structures with atomic accuracy even in cases in which no similar structure is known. Nobel Prize in Chemistry 2024.
194. Cappart, Q. et al. Combinatorial optimization and reasoning with graph neural networks. *J. Mach. Learn. Res.* **24**, 130 (2023).
195. Loukas, A. What graph neural networks cannot learn: depth vs width. Preprint at https://arXiv.org/abs/1907.03199 (2019).
196. Zhou, J. et al. Graph neural networks: a review of methods and applications. *AI Open* **1**, 57–81 (2020).
197. Prince, S. J. *Understanding Deep Learning* (MIT Press, 2023).
198. Xu, K. et al. What can neural networks reason about? In *International Conference on Learning Representations* https://openreview.net/pdf?id=rJxbJeHFPS (2020).
199. Loukas, A. How hard is to distinguish graphs with graph neural networks? In *Advances in Neural Information Processing Systems* Vol. 33 (eds Larochelle, H. et al.) 3465–3476 (Curran Associates, Inc., 2020).
200. Chen, Z., Chen, L., Villar, S. & Bruna, J. Can graph neural networks count substructures? In *Advances in Neural Information Processing Systems* Vol. 33 (eds Larochelle, H. et al.) 10383–10395 (Curran Associates, Inc., 2020).
201. Sato, R. A survey on the expressive power of graph neural networks. Preprint at https://arXiv.org/abs/2003.04078 (2020).
202. Garg, V., Jegelka, S. & Jaakkola, T. Generalization and representational limits of graph neural networks. In *Proc. 37th International Conference on Machine Learning* Vol. 119 (eds III, H. D. & Singh, A.) 3419–3430 (PMLR, 2020).
203. Barceló, P. et al. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations (ICLR 2020)* (Virtual conference, Ethiopia, 2020).
204. Selsam, D. et al. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations* (2019).
205. Bengio, Y., Lodi, A. & Prouvost, A. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res*. **290**, 405–421 (2021).
This paper surveys the recent attempts, both from the machine learning and operations research communities, at leveraging machine learning to solve combinatorial optimization problems.
206. Ozolins, E. et al. Goal-aware neural SAT solver. In *2022 International Joint Conference on Neural Networks (IJCNN)* https://doi.org/10.1109/IJCNN55064.2022.9892733 (IEEE, 2022).
207. Karalias, N. & Loukas, A. Erdős goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In *Advances in Neural Information Processing Systems* Vol. 33 (eds Larochelle, H. et al.) 6659-6672 (Curran Associates, Inc., 2020).
208. Morris, C. et al. Tudataset: a collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)* (2020).
209. Min, Y., Wenkel, F., Perlmutter, M. A. & Wolf, G. Can hybrid geometric scattering networks help solve the maximum clique problem? In *Proc. 36th International Conference on Neural Information Processing Systems*, NIPS '22 (Curran Associates Inc., Red Hook, NY, USA, 2024).
210. Gama, F., Ribeiro, A. & Bruna, J. Diffusion scattering transforms on graphs. In *International Conference on Learning Representations* (2019).
211. Zou, D. & Lerman, G. Graph convolutional neural networks via scattering. *Appl. Comput. Harmon. Anal.* **49**, 1046–1074 (2020).
212. Gao, F., Wolf, G. & Hirn, M. Geometric scattering for graph data analysis. In *Proc. 36th International Conference on Machine Learning. Proceedings of Machine Learning Research*. Vol. 97 (eds Chaudhuri, K. & Salakhutdinov, R.) 2122–2131 (PMLR, 2019).
213. Schuetz, M. J., Brubaker, J. K. & Katzgraber, H. G. Combinatorial optimization with physics-inspired graph neural networks. *Nat. Mach. Intell.* **4**, 367–377 (2022).
This publication stirred a debate whether the GNN-based method was adequately benchmarked against best prior methods.
214. Boettcher, S. Inability of a graph neural network heuristic to outperform greedy algorithms in solving combinatorial optimization problems. *Nat. Mach. Intell.* **5**, 24–25 (2023).
215. Angelini, M. C. & Ricci-Tersenghi, F. Modern graph neural networks do worse than classical greedy algorithms in solving combinatorial optimization problems like maximum independent set. *Nat. Mach. Intell.* **5**, 29–31 (2023).
216. Schuetz, M. J., Brubaker, J. K. & Katzgraber, H. G. Reply to: Modern graph neural networks do worse than classical greedy algorithms in solving combinatorial optimization problems like maximum independent set. *Nat. Mach. Intell.* **5**, 32–34 (2023).
217. Schuetz, M. J., Brubaker, J. K. & Katzgraber, H. G. Reply to: Inability of a graph neural network heuristic to outperform greedy algorithms in solving combinatorial optimization problems. *Nat. Mach. Intell.* **5**, 26–28 (2023).
218. Gamarnik, D. Barriers for the performance of graph neural networks (GNN) in discrete random structures. *Proc. Natl. Acad. Sci. USA* **120**, e2314092120 (2023).
219. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge Univ. Press, 2010).
220. Albash, T. & Lidar, D. A. Adiabatic quantum computation. *Rev. Mod. Phys.* **90**, 015002 (2018).
221. Chen, J. Review on quantum communication and quantum computation. *J. Phys. Conf. Ser.* **1865**, 022008 (2021).
222. Durr, C. & Hoyer, P. A quantum algorithm for finding the minimum. Preprint at https://arxiv.org/abs/quant-ph/9607014 (1996).
223. Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse ising model. *Phys. Rev. E* **58**, 5355–5363 (1998).
224. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. Preprint at https://arXiv.org/abs/1411.4028 (2014).
225. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**, 1484–1509 (1997).
226. Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proc. Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96* 212–219 (Association for Computing Machinery, New York, NY, USA, 1996).
227. Resch, S. & Karpuzcu, U. R. Quantum computing: an overview across the system stack. Preprint at https://arXiv.org/abs/1905.07240 (2019).
228. Harper, R., Flammia, S. T. & Wallman, J. J. Efficient learning of quantum noise. *Nat. Phys.* **16**, 1184–1188 (2020).
229. Martina, S., Buffoni, L., Gherardini, S. & Caruso, F. Learning the noise fingerprint of quantum devices. *Quantum Mach. Intell.* **4**, 8 (2022).
230. Suter, D. & Álvarez, G. A. Colloquium: Protecting quantum information against environmental noise. *Rev. Mod. Phys.* **88**, 041001 (2016).

231. Buffoni, L., Gherardini, S., Zambrini Cruzeiro, E. & Omar, Y. Third law of thermodynamics and the scaling of quantum computers. *Phys. Rev. Lett.* **129**, 150602 (2022).
232. Haverly, A. & López, S. Implementation of Grover's algorithm to solve the maximum clique problem. In *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* 441–446 (2021).
233. Bhaduri, A. S., Saha, A., Saha, B. & Chakrabarti, A. Robust quantum circuit for clique problem with intermediate qudits. *Nano Commun. Netw.* **37**, 100466 (2023).
234. Li, X., Wu, M., Chen, H. & Liu, Z. Algorithm for finding the maximum clique based on continuous time quantum walk. *Quantum Inf. Comput.* **21**, 59–79 (2021).
235. Li, X., Chen, X., Hu, S., Xu, J. & Liu, Z. A parameter-independent algorithm of finding maximum clique with Seidel continuous-time quantum walks. *iScience* **27**, 108953 (2024).
236. McGeoch, C. C. *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice* (Springer Cham, 2022).
237. Vinci, W., Albash, T., Paz-Silva, G., Hen, I. & Lidar, D. A. Quantum annealing correction with minor embedding. *Phys. Rev. A* **92**, 042310 (2015).
238. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).
239. Pattabiraman, B., Patwary, M. M. A., Gebremedhin, A. H., Liao, W. -k. & Choudhary, A. Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection. *Internet Math.* **11**, 421–448 (2015).
240. Kim, K., Kim, M., Park, J., Byun, A. & Ahn, J. Quantum computing dataset of maximum independent set problem on king lattice of over hundred Rydberg atoms. *Sci. Data* **11**, 111 (2024).
241. Munoz-Bauza, H. & Lidar, D. Scaling advantage in approximate optimization with quantum annealing. *Phys. Rev. Lett.* **134**, 160601 (2025).
242. Cazals, P. et al. Identifying hard native instances for the maximum-independent-set problem on neutral-atom quantum processors. *Phys. Rev. Appl.* **25**, 034085 (2026).
243. Pelofske, E. Comparing three generations of D-Wave quantum annealers for minor embedded combinatorial optimization problems. *Quantum Sci. Technol*. **10**, 025025 (2025). **Comparison between different generations of Quantum Annealers in solving various combinatorial optimization problems across different families of graphs.**
244. Pelofske, E., Hahn, G. & Djidjev, H. Solving large maximum clique problems on a quantum annealer. In *Quantum Technology and Optimization Problems: First International Workshop, QTOP 2019, Munich, Germany, March 18, 2019, Proceedings 1* 123–135 (Springer, 2019).
245. Naghsh, Z., Javad-Kalbasi, M. & Valaee, S. Digitally annealed solution for the maximum clique problem with critical application in cellular V2X. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)* 1–7 (IEEE, 2019).
246. Krpan, A., Povh, J. & Pucher, D. Quantum computing and the stable set problem. *Optim. Methods Softw.* **40**, 837–870 (2025).
247. Adams, C. S., Pritchard, J. D. & Shaffer, J. P. Rydberg atom quantum technologies. *J. Phys. B At. Mol. Opt. Phys.* **53**, 012002 (2019).
248. Ebadi, S. et al. Quantum optimization of maximum independent set using Rydberg atom arrays. *Science* **376**, 1209–1215 (2022). **Solution of MIS instances on a particular family of graphs showcasing a quantum speedup by using arrays of Rydberg atoms.**
249. Nguyen, M.-T. et al. Quantum optimization with arbitrary connectivity using Rydberg atom arrays. *PRX Quantum* **4**, 010316 (2023).
250. Vogel, J., Li, W., Mokhberi, A., Lesanovsky, I. & Schmidt-Kaler, F. Shuttling of Rydberg ions for fast entangling operations. *Phys. Rev. Lett.* **123**, 153603 (2019).
251. Aaronson, S. BQP and the polynomial hierarchy. In *Proc. Forty-Second ACM Symposium on Theory of Computing* 141–150 (2010).

Acknowledgements

R.M. is supported by #NEXTGENERATIONEU (NGEU) and funded by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), project MNESYS (PE0000006) "A Multiscale integrated approach to the study of the nervous system in health and disease" (DR. 1553 11.10.2022). L.B. was funded by PNRR MUR Project No. SOE0000098-ThermoQT financed by the European Union–Next Generation EU.

Author contributions

R.M. reviewed the heuristics and graph neural network solvers. L.B. reviewed the quantum algorithms. B.Z. reviewed exact solvers and constructed the benchmarks. All authors contributed to the writing of the manuscript.

Funding

Open access funding provided by HUN-REN Alfréd Rényi Institute of Mathematics.

Competing interests

The authors have no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Bogdan Zavalnij.

Peer review information *Communications Physics* thanks Fen Zuo and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Reprints and permissions information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2026