The Maximum Clique and Vertex Coloring

44

Oleksandra Yezerska and Sergiy Butenko

Contents

Abstract

In this chapter we review heuristic approaches for two classical and closely related problems of finding a maximum clique and an optimal vertex coloring. Both problems have a wide variety of practical applications, and due to their computational intractability, a significant effort has been focused on developing heuristic methods. This chapter discusses construction heuristics, local search strategies, and metaheuristics designed and/or adapted for the maximum clique and vertex coloring problems.

O. Yezerska · S. Butenko (✉)
Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX, USA
e-mail: yaleksa@tamu.edu; butenko@tamu.edu

R. Martí et al. (eds.), *Handbook of Heuristics*,
https://doi.org/10.1007/978-3-319-07124-4_47

Keywords
Maximum clique · vertex coloring · chromatic number

Introduction

Given a simple graph $G = (V, E)$, a *clique* is defined as a subset $C \subseteq V$ of mutually adjacent vertices. A graph is called *complete* if the set of its vertices forms a clique. A *maximal clique* is a clique that cannot be extended to a clique of larger size by simply adding a new vertex to this clique. The *maximum clique problem* asks for a clique of the largest cardinality in the graph. The size of a maximum clique is called the *clique number* of G and is usually denoted as $\omega(G)$.

An *independent set* (or *stable set*) is a subset $I \subseteq V$ of vertices with no edge between them. It is easy to see that a *clique* in G is an *independent set* in the *complement graph* of G, $\bar{G} = (V, \bar{E})$, where $\bar{E} = \{(i, j) : (i, j) \notin E \ \forall i, j \in V, i \neq j\}$, and vice versa. Finding a maximum clique in G is equivalent to finding a maximum independent set in $\bar{G}$. The cardinality of a maximum independent set in G is called the *independence* or *stability number* of G, denoted $\alpha(G)$, and $\alpha(\bar{G}) = \omega(G)$.

A *proper vertex coloring* (also referred to as a coloring for simplicity) is an assignment of a color (or a label) to each vertex in a graph such that no adjacent vertex has the same color. More formally, given a set of colors $\{1, 2, \ldots, k\}$, a *proper k-coloring* is a mapping $f : V \to \{1, 2, \ldots, k\}$, where $f(v_i) \neq f(v_j)$ $\forall (i, j) \in E$. Then the vertices with the same value of f belong to the same *color class*. The *chromatic number* of a graph G, denoted $\chi(G)$, is the minimum number of colors necessary to color G. An *optimal coloring* of a graph is a proper coloring that uses exactly $\chi(G)$ colors.

It is easy to see that the vertices that belong to the same color class form an independent set and the vertices of a clique all belong to different color classes. Then it follows that for a given graph G:

$$\chi(G) \geq |V|/\alpha(G),$$

and

$$\chi(G) \geq \omega(G).$$

The maximum clique and vertex coloring problems often arise in similar applications, where they play complementary roles. For example, let the vertices in the graph represent elements of a system and the edges between them – the incompatibility between those elements. Then a maximum clique will represent a set of mutually incompatible elements of the largest size in the system, whereas the color classes will correspond to sets of mutually compatible elements.

Both problems find a wide range of practical applications in various domains. Some of the earliest and best-known applications of the maximum clique problem

are in computer vision, experimental design, information retrieval, coding theory, fault diagnosis, social network analysis, and computational biochemistry and genomics, among other areas [32,150,182]. As for the coloring, it is commonly used for scheduling [125, 140], timetabling [173, 174, 178], frequency assignment [74, 176], circuit board testing [76], register allocation [38, 42], and, more recently, various problems in transportation [35, 177]. A discussion of both clique and vertex coloring applications in telecommunications is presented in [12].

The maximum clique and vertex coloring problems are both well-known NP-hard problems [75, 119] that are hard to approximate within a factor of $n^{1-\epsilon}$ for any $\epsilon > 0$, where $n = |V|$ [6,7,59,60,97,187]. Such intractability results, along with the practical interest, have led to a considerable effort in devising numerous heuristic approaches. Almost all known types of heuristic methods have been applied to these problems. Some of the early surveys on heuristic methods for the vertex coloring problem can be found in [151, 175] and more recent in [41, 71, 73, 132, 153], among which [41, 71] review just the local search methods and [73] focuses on the recent approaches. As for the maximum clique problem, the first extensive surveys date back to the 1990s [22, 150], while the most recent one [182] focuses mostly on local search techniques. Other brief discussions on heuristic approaches for the maximum clique problem can be found in [150, 153]. It should be noted that most of these surveys also discuss exact algorithms, many of which take advantage of heuristics to enhance their performance.

The remaining sections of this chapter are organized as follows. The most common construction heuristics for the maximum clique and vertex coloring problems are discussed in section "Construction Heuristics." In section "Local Search," we briefly outline the local search strategies used for both problems. Section "Metaheuristics" covers various metaheuristic approaches. It is followed by Sections "Heuristics Based on Continuous Formulations," "Other Heuristics," "Computational Results," and "Conclusion," respectively.

Construction Heuristics

Construction heuristics are used to build an initial feasible solution. The most intuitive and common construction heuristics are *sequential* ones, which recursively update the current, partial (and not necessarily feasible) solution one step at a time. Based on how the next step is selected, most of these heuristics can be classified as either greedy, random, or a combination of both. Greedy selection rules typically aim to ensure the highest level of flexibility (i.e., the largest possible "candidate set") in the future steps.

The most common sequential greedy heuristics for the maximum clique problem are *best in* and *worst out*, as named by Kopf et al. in [123]. Best in algorithms are those that start with an empty set as an initial solution and iteratively add a vertex with the largest degree among the candidate vertices, whereas worst out are the ones that start with a whole graph and recursively delete vertices with the lowest degree until the remaining graph is complete. The best in heuristic applied for finding the

maximal independent set yields a solution of size at least $\frac{|V|}{\delta+1}$, where $\delta = \frac{2|E|}{|V|}$ [58]. The best in heuristic was also shown to be "provably best" for the maximum clique problem in [115]. A heuristic is called provably best if no other polynomial-time algorithm can guarantee a better solution whenever one exists (assuming $\mathcal{P} \neq \mathcal{NP}$).

As for the vertex coloring problem, any heuristic that finds a coloring using at most $\Delta(G)$ colors (referred to as Brooks' coloring), where $\Delta(G)$ is the maximum vertex degree in G, is provably best [115]. According to Brooks' theorem [24], a simple, connected graph G that is neither complete nor an odd cycle satisfies $\chi(G) \leq \Delta(G)$. Brooks' coloring can be obtained using several heuristics [85, 91, 116, 118, 128, 163].

Many popular heuristics for the vertex coloring problem are based on sequential construction approaches. Sequential construction heuristics usually iteratively assign each vertex with the smallest feasible color such that no conflicts with already colored vertices occur. The order in which vertices are colored is either static and determined beforehand (preorder) or dynamic and updated on the fly. It is easy to see that the quality of the solution depends entirely on how the vertices are ordered and that there exists such vertex ordering that produces an optimal coloring [175]. Therefore, a lot of attention has been placed on studying and designing different ordering schemes.

Two examples of preorder sequential coloring heuristics are the *largest first* (LF) [173] and the *smallest last* (SL) [139]. LF sorts the vertices in decreasing order of their degree, and the ordering $(v_1, \ldots, v_n)$, produced in such manner, results in the coloring with at most $1 + \max_{1 \geq j \geq n} d_j(v_j)$ colors, where $d_j(v_j)$ is the degree of vertex v_j in the induced subgraph $G_j = G[\{v_1, \ldots, v_j\}]$. SL is similar to LF, but it sorts the vertices in the reverse order. First, a vertex of the minimum degree in graph G is selected and labeled v_n. Then, iteratively for each $j = n-1, \ldots, 1$, a vertex of the minimum degree in G_j is selected and labeled v_j.

A notable sequential coloring approach that does not preorder vertices is the DSATUR [23] heuristic. A decision on what vertex to color next is based on the value of the vertex *saturation degree* – a number of the differently colored neighbors of this vertex. A vertex with the maximum saturation degree is selected. Another example of a heuristic that does not preorder vertices is the *recursive largest first* (RLF) [125]. This heuristic generates a color class one at a time and does not proceed to another color class until no more vertices can be assigned to the current class. Such an assignment is implemented in the following manner. When a new color class is generated, set U_1 contains all uncolored vertices, and set U_2 is empty. Iteratively, a vertex $v \in U_1$ is selected, assigned to the current color class, and removed from U_1. If v has any neighbors in U_1, they are removed from U_1 and placed in U_2. The vertex assignment proceeds while U_1 is not empty. The very first vertex to be assigned to the new color class is the one with the maximum degree in $G[U_1]$, and all the rest are the ones with the maximum degree in $G[U_2]$. When U_1 becomes empty, the next color class is generated, and the process repeats.

In another study [20], a maximal independent set is iteratively extracted from the set of uncolored vertices, and all the vertices forming this independent set are

assigned the same color. The process repeats until the whole graph is colored. Similar technique is used in the algorithm called XRLF [114], which is a version of RLF [125]. Several independent sets are first constructed, and then the one whose removal yields a residual graph with the smallest edge density is selected as a new color class. This process is repeated until a threshold number of uncolored vertices are left, which are then colored using an exhaustive search. Independent set extraction approach has been used to construct initial colorings within various metaheuristic frameworks [39,66,73,99,142]. Recently, a few enhanced approaches based on independent set extraction were proposed. Namely, in [179], the authors suggest to preprocess large graphs by iteratively extracting a maximal number of pairwise disjoint independent sets of the same size. Such strategy was shown to assign more vertices to the same number of color classes than that with a one-at-a-time independent set extraction. This yields smaller in size residual graphs of uncolored vertices, which are easier to color. In [96, 181], the authors observe that an independent set extracted may not necessarily define a color class in an optimal coloring and that removing such sets during preprocessing may prevent one from reaching an optimal solution. To mitigate this drawback, the authors of [96,181] suggested a framework that reconsiders a certain number of the removed independent sets and allows some of the vertices forming those sets to change a color. Different strategies on how many and which extracted sets to reexamine were studied, and the experiments on some large and hard graph instances reported in [96, 181] demonstrate an encouraging performance of the proposed approach.

To improve the quality of solutions generated by the greedy sequential methods, different techniques ranging from randomization, multiple restarts, and local search (section "Local Search") to sophisticated metaheuristic frameworks (section "Metaheuristics") can be used. We will cover here a few simple strategies and discuss the rest in the later sections.

For the maximum clique problem, Jagota et al. [109] proposed several enhancements to a basic greedy sequential heuristic, which include randomization, multiple restarts, and an adaptive mechanism. A randomization is embodied by a probabilistic greedy vertex selection rule. Three kinds of adaptation are studied: an update of the initial state (AI), an update of the probability distribution used by a selection rule (AW), and no adaptation at all (NA). The corresponding updates take place at each restart and are based on the information obtained in the previous restart. NA heuristic was shown to perform poorly compared to AI and AW. Grosso et al. [87] used similar techniques and developed a two-phase heuristic, called a *deep adaptive greedy search* (DAGS). In the first phase of DAGS, the modified variant of a sequential greedy heuristic is applied for all the vertices in the graph. The modification consists of allowing to swap already added vertices with the better candidates. The second phase is used for diversification purposes. It is a restart-adaptive greedy heuristic performed on a set of vertices that appear less frequently in the cliques obtained during the first phase. Each vertex is assigned a weight associated with its quality as a potential candidate for a solution. Such weight is updated at each restart by means of a learning-like mechanism.

For the vertex coloring problem, an improvement procedure called an *iterative greedy* (IG) heuristic was proposed in [46, 47]. It is based on the observation that given a feasible coloring and an ordering in which vertices belonging to the same color class are grouped together, applying a sequential greedy heuristic to such an ordering will produce a solution at least as good as the previous one.

Local Search

Local search is a technique of improving a current solution by exploring its local neighborhood and moving to better neighboring solutions until no more improvement can be achieved. A local neighborhood of a solution is defined by a *neighboring function* which can also be thought of as a distance or a difference between neighboring solutions. Local search methods vary based on the definition of a neighboring function and a corresponding local neighborhood, as well as the choice of a search space and a function that evaluates the quality of a solution.

As suggested in [182], local search strategies for the maximum clique problem can be categorized into *legal* and *k-fixed penalty* ones. A legal strategy is a traditional local search, where a search space consists of legal (feasible) solutions, cliques; an evaluation function is the size of a clique; and a neighboring function is a list of vertex operations to be performed to move from one clique to another. The most common neighboring functions are (a, b)-interchanges, where a is the number of vertices removed from the current solution and b is the number of added vertices, with a usually smaller than b to ensure that such interchange increases the size of the solution by $(b - a)$.

A k-fixed penalty strategy, on the other hand, can be thought of as a technique that solves the decision version of the problem that answers the question "is there a clique of size k?" This type of a local search deals with infeasible solutions, sets that are not necessarily cliques. An evaluation function is defined as the number of edges in the set. By moving to sets with larger number of edges, a local search attempts to eventually reach a feasible solution – a clique. For this type of a local search, a (1,1)-interchange is usually used, where a vertex to be removed is usually the one with the smallest number of neighbors in the solution and a vertex to be added is the one adjacent to the largest number of solution members. Such a technique does not guarantee finding a feasible solution; therefore, it is usually used as a part of a more sophisticated local search framework (section "Metaheuristics").

For the vertex coloring problem, the local search strategies can also be categorized into legal and k-fixed ones. Moreover, as discussed in [71], they can be further grouped into *legal*, *k-fixed partial legal*, *penalty*, and *k-fixed penalty* ones. Here, the legal and the k-fixed penalty strategies, similarly to the local search strategies for the maximum clique problem, are the local searches on either feasible (proper) colorings or infeasible k-coloring, respectively. For the legal strategy, the local neighborhood is explored in an attempt of finding a feasible coloring with fewer number of colors. An infeasible coloring is such that contains at least one pair of adjacent vertices assigned the same color. An edge connecting a pair of such

vertices is referred to as *conflicting*. Then for this strategy, a local search aims at performing moves resulting in the minimization of the number of conflicting edges in the solution.

Under a k-fixed partial legal strategy, the search space contains partial feasible k-colorings. A partial feasible k-coloring corresponds to a coloring of a subset of vertices $V' \subset V$ that uses k colors and is feasible with respect to already colored vertices in V'. The rest of the vertices in $V \setminus V'$ are not yet colored. The local neighborhood is explored in an attempt to find a complete feasible k-coloring. For a pure penalty strategy, the search space contains infeasible colorings, and the number of colors used is unfixed. The local search under this strategy tries to detect a feasible coloring with the minimum number of colors used. The quality function usually combines both the number of conflicting edges and the number of colors used.

The most common neighboring functions for coloring are also interchanges [139]. They are performed by moving vertices from one color class to another, in an attempt of emptying one of the classes and, thus, reducing the number of colors used. Under the penalty and k-fixed penalty strategies, such interchanges are allowed to result in infeasible solutions.

Local search approaches also vary based on the rules of determining when to move to a better solution. By the *best improvement strategy*, the entire local neighborhood is explored, and the best neighbor is selected as a new solution, whereas the *first improvement strategy* updates the current solution with the first found neighbor of a better quality. However, with either of these strategies, a local search tends to get trapped in local optima of poor quality. To overcome this drawback and explore more regions of a search space, different techniques, e.g., allowing non-improving moves and recording the search process to direct the future exploration, have been devised and will be discussed in the following section.

Metaheuristics

In this section, we will review the most common metaheuristics for the maximum clique and vertex coloring problems. The metaheuristics can be classified with respect to various criteria [19]. Based on the number of solutions maintained at the same time, the algorithms can be divided into single-point search ones, encompassing most local search-based heuristics (section "Local Search-Based Methods"), and population-based ones (section "Population-Based Methods"), which deal with a pool of solutions at a time and perform evolution-like search processes.

Local Search-Based Methods

The most successful techniques to enhance local search have proven to be the ones that allow non-improving moves in the process of local neighborhood exploration. One of them, *simulated annealing* (section "Simulated Annealing"), allows moving to a worse solution with a certain probability, while another one, *tabu search*

(section "Tabu Search"), records the moves and forbids (tabu) their future usage for a certain number of iterations to prevent cycling. Other successful local search-based methods have been devised and will be discussed in section "Other Local Search-Based Methods".

Simulated Annealing

Simulated annealing (SA) method is a randomized neighborhood search introduced by Kirkpatrick et al. [121] and covered in a separate chapter of this book. SA is analogous to a physical annealing – a process of heating up a solid material and slowly cooling it down in order to obtain a low energy configuration. The main idea of SA is to iteratively select random candidate solution in the local neighborhood and, if it is of better quality, move to the candidate solution; otherwise perform the move with a certain probability of acceptance. Such probability is usually defined as $\exp(-\Delta)/T$, where Δ is a difference between the objective function values of the new and the current solutions and T is a parameter called the *temperature*. As the heuristic progresses, the probability is being decreased, ensuring fewer random walks and directing the search toward improvement. The rate at which the probability decreases is usually referred to as a "cooling schedule." The cooling schedule that follows the logarithmic law guarantees the convergence of the algorithm to a global optimum [19]; however, it may result in an exponential running time. Hence, faster cooling schedules are usually used in practice.

The most influential papers devoted to application of SA to the maximum clique problem date back in the late 1980s and mid-1990s and are all covered in [22]. Since, to the best of our knowledge, no novel SA methods for the maximum clique problem have been designed since then, we will follow [22] and briefly review here the same studies.

An application of SA to the maximum clique (independent set) problem was first described by Aarts et al. in 1988 [1]. Without providing any experimental results, they proposed a *penalty function* method (k-fixed penalty strategy), for which a possible solution can be any set, not necessarily independent, and the quality of the solution is determined by the function $f(V') = |V'| - \lambda|E'|$, where V' and E' are the vertex and edge sets of the solution and λ is a certain weighting factor. A few years later, Jerrum et al. [111] have theoretically proven a poor performance of SA application to the maximum clique problem. In particular, they studied a variant of SA with the temperature parameter fixed and a solution space consisting of legal cliques (legal strategy). Feo et al. [64] implemented SA for the maximum independent set problem utilizing the penalty method and observed that it was inferior to the greedy randomized adaptive search procedure (GRASP) they proposed. However, the computational experiments presented by Homer et al. [104, 113] in the Second DIMACS Implementation Challenge showed that SA can be very effective. In their work, Homer et al. used the idea of penalty method described by Aarts and implemented it for the maximum clique problem. The results of the experiments on large graphs, reported in [104, 113], were quite promising, and SA was ranked one of the best heuristics in the Second DIMACS Implementation Challenge.

One of the first attempts to apply SA to the vertex coloring problem dates back to 1987, when Chams et al. [39] considered the following variant of the method. The neighborhood structure consists of k-colorings that are not necessarily legal (k-fixed penalty strategy), and the objective function to be minimized is defined by the number of conflicting edges. According to the experimental results reported in [39], the pure SA did not perform as strongly as when it was combined with other methods. In particular, when SA was combined with RLF [23], where RLF was used to generate color classes and SA to color the rest of the uncolored vertices, the performance of such combined method dominated all other methods tested in the paper (DSATUR and RLF). In 1991, Johnson et al. [114] conducted a detailed study of different schemes of SA adopted to the vertex coloring problem. The first scheme tested is a penalty function method, where the neighborhood structure is defined by infeasible solutions (penalty strategy), i.e., colorings with conflicting edges. The number of colors used in a particular solution is not fixed. Let C_i denote a color class, let E_i be an edge set in the subgraph induced by C_i ($i = 1, \ldots, k$), and let $\Pi = C_1, \ldots, C_k$ ($1 \leq k \leq |V|$) be a solution. Then the cost (penalty) function associated with Π is defined as follows:

$$f(\Pi) = \sum_{i=1}^{k} 2|C_i| \cdot |E_i| - \sum_{i=1}^{k} |C_i|^2.$$

The first component of $f(\Pi)$ favors independent sets, while the second one favors large color classes. Minimization of $f(\Pi)$ results in eliminating the conflicting edges and reaching a legal coloring and also, as the authors argued, in potentially reducing the number of color classes as a side effect. The second scheme is by Morgenstern and Shapiro [143], and it utilizes so-called *Kempe chains*, which are defined in [114] as the connected components of the union of two disjoint independent sets. In this method, the same cost function as in the penalty function method is retained, but the search space contains legal k-colorings. The new candidate solution is generated in the following way. Two color classes C_i and C_j ($i \neq j$) are picked at random, and a Kempe chain defined by a set $H \subset C_i \cup C_j$ is constructed. Next C_i is replaced by $C_i \ \Delta \ H$ and C_j – by $C_j \ \Delta \ H$, where $X \ \Delta \ Y$ denotes a symmetric difference $(X - Y) \cup (Y - X)$ between X and Y. The third scheme uses the same strategy as in [39], where the number k of colors is fixed and the objective is to minimize the number of conflicting edges in an attempt to find a feasible k-coloring. According to the results of the experiments reported by Johnson et al., none of the three methods was found dominant over the other.

A simplified version of SA with the temperature parameter remaining fixed was used by Morgenstern [113, 142] for the Second DIMACS Implementation Challenge. In this paper, a new neighborhood structure, called *impasse neighborhood*, was defined. It turned out to be very effective and many approaches have used it thereafter. This neighborhood structure operates with a fixed number k of colors and aims to improve a partial coloring, in which not all vertices are colored, to a complete coloring with k colors. A feasible solution is a partition of the set of

vertices into $k + 1$ subsets $\{V_1, \ldots, V_k, V_{k+1}\}$, where $V_1, \ldots, V_k$ are independent sets representing a partial k-coloring and V_{k+1} is the set of uncolored vertices that needs to eventually be emptied in order to obtain a proper k-coloring of all vertices. A neighbor of a solution is obtained by moving a vertex v from V_{k+1} to one of the k independent sets, say V_i, and then moving to V_{k+1} all vertices from V_i that are adjacent to v.

Tabu Search

Tabu search was developed by Glover [81] and, independently, by Hansen and Jaumard [93]. See the ▸ Chap. 25, "Tabu Search" for more details. It is a modified local search which allows performing moves yielding worse quality solutions, thus diversifying the search path. The algorithm uses the so-called *tabu lists* to store certain information about the previously visited solutions to forbid future local search moves that can result in cycling. The length of the tabu lists, called *tabu tenures*, allows to control the level of diversification and intensification of the search. Sometimes the tabu restriction is relaxed if a solution meets certain *aspiration level* condition.

In 1989, Friden et al. [68] used tabu search ideas to construct a heuristic called STABULUS for finding stable (independent) sets. STABULUS is based on the k-fixed penalty local search strategy (see section "Local Search"), where the search space consists of the sets of size k, and the algorithm tries to minimize the number of edges in the set. To find an independent set of the largest cardinality, STABULUS is applied iteratively, incrementing k each time an independent set of size k is detected. Three tabu lists were used in STABULUS, one for storing visited solutions, and the other two contained added and deleted vertices.

A few years later, Gendreau et al. [78] proposed two simple variants of tabu search for the maximum clique problem, deterministic and probabilistic, where in the former, at each iteration, the entire set of the solution neighbors is explored and, in the latter, the sampling of the neighborhood is applied. In both variants the legal local search strategy was exploited, where the search space contains the feasible cliques and the objective is to find a clique of the maximum size. In the deterministic approach, it is allowed to remove only one vertex per iteration, while in the probabilistic it is a few at a time. Later these heuristics were enhanced with the diversification strategies [166], such as greedy restart and continuous diversification. The first one is a very simple approach of restarting the algorithm with the new solution after a certain number of iterations without improvement have been performed. A new solution is constructed from vertices that have not been visited yet or have been visited less frequently. The continuous diversification works in "on-the-fly" mode. It guides the search path by evaluating each vertex addition/removal using the information on how frequently the vertex has been added to the solution and how long it has stayed in the solution. Both diversification techniques and their combination have proven to be beneficial if used along with the simple basic search. The more efficient and sophisticated the search is, the less improvement the diversification is able to add to it. It also did not appear to work

well with the probabilistic version of the tabu search heuristic. The heuristics were tested on DIMACS instances, and the results were reported in [167].

A similar idea of prohibiting the future moves in order to prevent cycling was used in reactive local search (RLS) [14]. However, contrary to the traditional tabu search, where certain parameters (tabu tenure, depth of the search, etc.) are determined by the user through numerous preliminary experiments and are usually sensitive to certain specific structures of the instances, RLS determines all of this information dynamically within its execution. The length of tabu lists is adjusted based on previously stored information on how often the algorithm cycles, etc. Also, the algorithm is equipped with a memory-influenced restart procedure to provide additional long-term diversification. RLS showed excellent performance on DIMACS instances and is considered one of the best local search-based heuristics for the maximum clique problem.

Recent efforts of applying tabu search to the maximum clique (independent set) problem include works by Wu et al. [112, 180, 183]. In [180], the authors developed an adaptive multistart tabu search approach that uses a k-fixed penalty local search strategy. The exploration of a search space is performed by moves that swap a vertex from the current solution with a vertex from the solution constrained neighborhood. The algorithm employs a restart strategy, under which a new solution is constructed from the vertices that were less frequently used throughout the algorithm, that way visiting unexplored search space regions. The algorithm showed excellent performance when compared against several other state-of-the-art algorithms. The study of the relationship between the number of restarts and the quality of the evaluation function performed by the authors suggested that for structured instances, more frequent restarts yield better results, while the reverse holds for random graphs.

For the vertex coloring problem, tabu search techniques were first applied by Hertz and de Werra [99]. In their proposed algorithm, named TABUCOL, the search space consists of the complete k-colorings, where k is fixed. Some colorings might have conflicting edges, and, hence, the algorithm's goal is to move to the solutions with smaller number of conflicting edges by changing the color assignment of the endpoints of conflicting edges. The tabu lists store these color assignments. The algorithm uses the same local search strategy as in [39] but yields better results than those obtained by SA in [39].

Recently, Blöchliger and Zufferey [18] designed several variations of the tabu search framework for the vertex coloring problem. The main ingredients they used were the partial k-coloring local search strategy and reactive tabu tenure, adjusted based on the fluctuations of the objective function.

Tabu search has also been applied to both the maximum clique and vertex coloring problems as a part of more complex metaheuristics, such as evolutionary algorithms, yielding the so-called *hybrid algorithms* (discussed in section "Population-Based Methods").

Other Local Search-Based Methods

The greedy randomized adaptive search procedure (GRASP) [62, 63], described in the ▸ Chap. 16, "GRASP" in this volume, is a simple metaheuristic that combines

a constructive heuristic, a local search, and a restart policy. Its construction part uses both greedy and random selection rules. More precisely, the initial solution is built by randomly selecting a vertex from a so-called *restricted candidate list* (RCL), which consists of a certain number of the best candidates. This number ranges between 1 and the size of the candidate list, where the closer it is to 1, the greedier the selection rule is, and the closer it is to the candidate list size, the more randomness is involved. Every time a solution is constructed, an attempt to improve it is made using an appropriate local search technique (see section "Local Search"). The algorithm restarts a certain number of times, and the best found solution is recorded. GRASP has been successfully applied to the maximum clique (maximum independent set) [2, 64, 161] and the vertex coloring problems [124].

Variable neighborhood search (VNS) [94, 141] is a metaheuristic that systematically explores different neighborhood structures of the same problem in the search for a better solution. This method is introduced in the ▸ Chap. 26, "Variable Neighborhood Search". VNS approach was applied to the maximum clique problem in [95]. For the vertex coloring, an algorithm based on VNS was proposed in [8]. It uses more than ten different neighborhoods and utilizes tabu search techniques. An extension of VNS, called *variable space search* (VSS) was proposed for the vertex coloring problem by Hertz et al. in [101]. In VSS, when the search is trapped in a local optimum, the entire search space, including the neighborhood searched and the objective function, is changed. The proposed algorithm, called VSS-Col, explores three different search spaces for the vertex coloring problem and yields excellent results. An *iterated local search* (see the ▸ Chap. 19, "Iterated Local Search" in this handbook for more information), which upon reaching a local optimum perturbs it so that the local search can keep going, was applied to the vertex coloring problem in [40, 148]. In [40], the authors analyzed the performance of different types of solution perturbations.

An algorithm based on *variable depth search* for the maximum clique problem, based on k-*opt* local search (KLS), was proposed by Katayama et al. [120]. The k-opt search is essentially an (a, b)-interchange with $a > b$ and $a + b = k$, where a vertices are removed from the current clique and b vertices from the local neighborhood are added to the solution yielding a clique of a larger size. The value k is determined dynamically throughout the execution of the algorithm. The reported results on selected DIMACS instances show the algorithm's competitiveness with RLS [14].

Pullan and Hoos [159] introduced a dynamic local search (DLS) algorithm for the maximum clique problem. DLS is based on alternating between a clique expansion phase, during which vertices are added to the current clique, and plateau search, during which vertices of the current clique are swapped with vertices outside of the current clique. The selection of vertices is based on the penalties assigned to them dynamically during the search in a similar way used in DAGS [87]. When no more expansion or swapping is possible, a perturbation mechanism is applied to overcome search stagnation. The algorithm achieves substantial improvements over some state-of-the-art algorithms on many DIMACS instances. However, the algorithm has a disadvantage of being sensitive to the penalty delay parameter,

which controls the frequency at which vertex penalties are decreased and needs to be determined beforehand by preliminary experiments and thorough calibration. This issue has been addressed in [158], where a modified algorithm, called a Phased Local Search (PLS), was proposed. Similarly to DLS [159], the algorithm is a combination of a clique expansion and a plateau search. Moreover, it operates three sub-algorithms, which differ in their vertex selection rule: random selection, random selection within vertex degree, and random selection within vertex penalties. In the latter one, the parameter responsible for the frequency of penalty decreases does not have to be fixed and defined externally as in DLS but is modified adaptively. PLS has shown to have a comparable and sometimes improved performance to DLS. Grosso et al. [88] performed a detailed computational study of the main components of DLS, such as usage of vertex penalties as a selection rule, plateau search, etc., in an attempt to design simple and efficient iterated local search techniques for the maximum clique problem. Another study that analyzes the key ingredients of RLS and DLS was performed by Battiti et al. in [13] and yielded several enhanced modifications of the mentioned algorithms.

Recently, Pullan et al. [160] introduced a parallelized hyper-heuristic algorithm for the maximum clique problem, called a cooperating local search (CLS). Hyper-heuristics [29] are heuristic search methods that manage the low-level heuristics. In particular, CLS is a methodology that controls the four PLS-based heuristics and dynamically reconfigures their allocation to cores, based on information retrieved from a trial in order to ensure the appropriateness of selected heuristic for a particular instance. In [5], Andrade et al. introduce efficient implementations of $(1, 2)$ and $(2, 3)$-interchange legal strategy local searches for the maximum independent set problem. These two local searches are integrated into an iterated local search metaheuristic, which according to the results reported in [5] yield great results, especially on large and difficult instances. Benlic and Hao [15] devised an algorithm, called breakout local search (BLS). BLS is an iterated local search, which, when a local optimum is discovered, applies a certain diversification strategy to perturb the current solution, so it can be used as a starting point to explore another region of the search space. The magnitude of perturbation is adapted dynamically according to the current search state. The results of the experiments showed that BLS competes favorably with RLS [14], VNS [95], and PLS [158].

Population-Based Methods

As opposed to local search algorithms, where one solution is maintained at each iteration, the population-based algorithms deal with several solutions (a *population*) at a time. The new population is produced by certain interactions between the members of a current population. Namely, the population generation techniques in *evolutionary* and *genetic algorithms* (section "Evolutionary and Genetic Algorithms") are based on imitating the mechanisms of evolution, whereas in *ant colony algorithms* (section "Ant Colony Optimization"), the solution construction is inspired by the behavior of ants seeking paths to food sources.

Evolutionary and Genetic Algorithms

Evolutionary and genetic algorithms [82, 103] (see also the corresponding chapters in this volume) are inspired by the evolution and natural selection and are based on the following evolution principles: "the fittest survives," "two fit parents will have even fitter offsprings," and mutation. In the genetic algorithms, these principles are embodied with the help of respective mechanisms: *reproduction*, *crossover*, and *mutation* operators. At each iteration (in evolutionary terminology – *generation*) of an algorithm, the population of solutions (*individuals*) is subjected to all or some of those operators. Reproduction operator selects the "fittest" solutions, where the fitness level is represented by a *fitness function* (or objective function in optimization terms). Then, with the help of the crossover operator, the "offspring" solutions are produced as a result of merging one or several parts of one parent solution with one or several parts of another parent solution. The mutation operator changes a small part of a solution to provide randomization.

For the maximum clique problem, the most intuitive way to encode a solution is to use a binary array of the size equal to the cardinality of the given graph's vertex set, where an entry contains 1, if the corresponding vertex is included in the current solution, and 0, otherwise. The definition of the solution fitness function can vary. A fairly simple fitness function assesses the size of a vertex subset if it is a clique, or equals 0, otherwise. It can also be a little more complex. For example, it can combine several terms, such as the density of the induced subgraph and its size. It can also include a term to penalize the infeasible solutions.

The early work on adapting genetic algorithms to the maximum clique problem is covered in great detail in [22]. We will review in the following the same studies and also discuss several recent advancements.

One of the earliest studies on the effectiveness of genetic algorithms at solving the maximum clique problem was done by Carter et al. [37]. The authors showed that the pure genetic algorithm performs poorly and designed several modifications, including an annealed fitness function and the modified crossover operators, to improve its performance. However, their results were still not satisfactory and led to a conclusion that genetic algorithms have to be significantly customized to be able to perform well and that they are very computationally expensive. Their later studies in [152] report that genetic algorithms are ineffective for solving combinatorial optimization problems and are inferior to other heuristics, such as simulated annealing. However, in another study [9], Back et al. showed quite the contrary results. They designed a genetic algorithm called GENEsYs with a fitness function which included a component for penalizing infeasible solutions. The performance of GENEsYs for the maximum independent set problem on graphs with up to 200 vertices was promising and suggested that with the right fitness function, genetic algorithms are capable of yielding satisfactory results. In another approach, Murthy et al. [146] introduced a new crossover mechanism, called a *partial copy crossover*, and suggested to split a mutation operator into a *deletion* and *addition* operators. The results presented in the paper are based on experiments with small graphs (up to 50 vertices), making it difficult to evaluate the algorithm's effectiveness.

In [67], Foster and Soule developed two variations of genetic algorithms for the maximum clique problem. One of them uses a different problem-encoding approach, referred to as a *grouping representation*, which instead of encoding each solution as a binary string, uses an array that stores information about all the solutions in the current population. Such encoding scheme showed to increase effectiveness of the crossover operation. The other variant of the algorithm uses the time-weighted fitness function, which showed to improve the algorithm's performance. Hifi [102] adopted a genetic algorithm for solving the maximum weighted independent set problem. The necessary algorithm modifications included an introduction of a so-called *two-fusion crossover operator* and a replacement of a mutation operator by a *heuristic-feasibility operator*. The two-fusion crossover operator takes into account both the structure and the fitness of two parent solutions and produces two children, while the heuristic-feasibility operator transforms infeasible solutions into feasible ones. The main contribution of the designed approach is that it is easily parallelizable.

Aggarwal et al. [4] designed an optimized crossover mechanism (inspired by a heuristic CLIQMERGE proposed in [10] by Ballas and Niehaus) and applied it to the maximum independent set problem. The new crossover operator exploits the structure of the solution rather than its encoding. More precisely, when a crossover operator is applied to two solutions from the population, one of the new solutions is generated in such a way that it has the best fitness function value, while the other one is constructed in a way that ensures the diversity of the search space. Promising results of the proposed mechanism inspired authors of CLIQMERGE, Balas and Niehaus, to examine this strategy even further [11]. The authors also considered a different way of population replacement called a *steady-state replacement* and a different selection method. The results of the experiments reported in [11] were even more encouraging than those by Aggarwal et al. in [4].

Since most of pure genetic algorithms have shown to be not very effective for solving hard combinatorial optimization problems, they are usually enhanced by incorporating different techniques. Bui and Eppley [26] designed a hybrid strategy with vertex preordering phase and a local optimization at each iteration of the genetic algorithm. The authors claim that the performance of their hybrid algorithm is comparable to a continuous-based approach by Gibbons et al. [79] but inferior to tabu search- and simulated annealing-based approaches. Fleurent et al. [65] used tabu search and other techniques as alternative mutation operators. The performance of their approach was satisfactory with respect to the quality of the obtained solutions but rather disappointing with respect to the computational effort. Marchiori [136] developed an approach that combines a genetic algorithm and a greedy heuristic, referred to as HGA. At each iteration of genetic algorithm, the greedy heuristic is applied to each member of the population to extract maximal cliques. That way, the genetic part of the approach provides exploration of the search space, while the greedy heuristic provides exploitation. Despite its simplicity, the approach has shown very good results [136, 137]. Zhang et al. [186] developed an approach called EA/G which uses similar techniques as HGA but also incorporates a guided mutation operator. The results of the experiments indicated that the algorithm

was superior to HGA. Singh and Gupta [162] designed a framework which consists of two phases: generation of cliques by a steady-state genetic algorithm and extending them to maximal cliques by a heuristic which combines a randomized sequential greedy approach and the exact algorithm of Carraghan and Pardalos [36]. The results of the experiments show that the algorithm outperformed three other evolutionary algorithms, of Balas and Niehaus [11], Marchiori [137], and Fenet and Solnon [61], but its performance is inferior to that of one of the most effective local search-based techniques (RLS) of Battiti and Protasi [14].

More recently, Guturu and Dantu [90] designed an approach that combines an impatient evolutionary algorithm and a probabilistic tabu search. Brunato and Battiti [25] presented a hybrid algorithm similar to EA/G by Zhang et al. in [186]. In the proposed algorithm, which they refer to as R-EVO, the new solutions are initialized according to an estimated distribution which is based on knowledge extracted from the previous generations of the population. The solutions are then evolved through a simplified version of RLS technique.

Despite such a significant effort in studying and designing different variations of evolutionary and genetic algorithms, these approaches are still considered to be far less effective for the maximum clique problem than the local search-based ones. Perhaps, their inability to compete with the simple local search techniques lies in the fact that there is no intuitive relationship between the crossover operator and a discovery of improved cliques, and no such a meaningful operator has been devised yet [182].

For the vertex coloring problem, the solutions are usually encoded either as permutations of the vertices and referred to as *order-based encoding* or as color partitions and referred to as *direct encoding*. The direct encoding can further be categorized into the *assignment based* and *partition based*. Under the former one, a solution is represented as an array, where each entry is associated with a certain vertex and contains an index of the class the corresponding vertex belongs to. The latter one, on the other hand, represents a solution as a partition of the vertices into the color classes. Even though more satisfactory results have been achieved using direct encoding techniques, the order-based encoding has a powerful advantage of always corresponding to a feasible solution. Hence, it has been somewhat exploited as well [48, 57, 145].

One of the first attempts to apply evolutionary algorithms for vertex coloring problem is by Davis [48]. The author used order-based encoding, and a solution was evaluated using a greedy sequential heuristic, which colors the vertices in the order defined by the permutation. The algorithm yielded unsatisfactory results, as was also shown in [66]. All of the later efforts of applying evolutionary algorithms to the vertex coloring problem involved the incorporation of other techniques, such as local search. The pioneers of this direction were Costa et al. [44] and Fleurent and Ferland [66], replacing the mutation operator by a simple descent method in the former and tabu search in the latter, respectively. In both papers the uniform crossover operator or its slightly enhanced modification [66] is used. The algorithms yielded slightly better results than their pure counterparts; however, the importance of designing better crossover operators was concluded in [66].

In later developments, the focus shifted to designing more advanced crossover operators tailored to the specifics of the problem of interest. More precisely, Dorne and Hao [52] used a specialized crossover based on the notion of the union of independent sets, which, with the combination of tabu search, led to a simple yet very powerful algorithm. Galinier and Hao [70] devised a new class of crossover mechanisms based on the partition of vertices into color classes rather than assignment of colors to vertices. The crossover operator called *greedy partition crossover* (GPX) transmits the subsets of color classes from parent solutions to the offspring solution. The power of this operator lies in its ability to retain some of each parent's structure in the offspring solution. The hybrid algorithm that uses this crossover operator and a tabu search was tested on large and difficult DIMACS instances and showed to be one of the best performing compared to other coloring algorithms [70]. A few years later, Glass and Prügel-Bennett [80] examined the hybrid algorithm of Galinier and Hao [70] by replacing the tabu search part of the algorithm with a steepest descent. The results of the experiments showed that the algorithm still remained powerful, which led to a conclusion that its success is due to the crossover operator. Hamiez and Hao [92] presented a *scatter search algorithm* devised for the vertex coloring problem. A scatter search is a subclass of evolutionary approaches, where replacements of the population are based not only on the improvement of the fitness function but also on the improvement of the population diversity; see the corresponding chapter in this handbook for more detail. The algorithm proposed in [92] was able to obtain results similar to the best-known approaches, but it was more expensive computationally.

More recently, Galinier et al. [72] proposed an *adaptive memory algorithm*, called AMACOL, where portions of solutions (color classes) are stored in a central memory and are then used to build new solutions. The algorithm uses a tabu search and yields results comparable to those obtained by an algorithm proposed in [70]. Malaguti et al. [133] designed a two-phase metaheuristic algorithm, called MMT, for the vertex coloring problem. The first phase is a combination of an evolutionary algorithm with a crossover similar to GPX from [70] and a tabu search with an impasse neighborhood, proposed in [142]. The second phase of MMT is a post-optimization procedure based on the set covering formulation of the problem. The algorithm yields promising results.

Ant Colony Optimization

Ant colony optimization, discussed in more detail in a separate chapter of this book, was originally introduced by Dorigo [51] and was inspired by the way the ants use pheromone to communicate with each other and search for better paths to a source of food. As they search for food, each ant leaves a pheromone trail. All subsequent ants will follow the path with stronger pheromone. Since the pheromone evaporates over time, it will last longer on better (shorter) paths. This way, these paths will be followed by more ants and will subsequently have more pheromone laid. Eventually all the ants will follow one single best path. The main idea of ant colony optimization algorithms is to mimic this process. Each ant is thought of as a constructive heuristic,

which builds a solution step by step using a greedy force and a *trail* information on the history of the search obtained from other ants.

First attempt to investigate ant colony capabilities to tackle the maximum clique problem was by Fenet and Solnon [61]. They introduced an algorithm, called Ant-Clique, that generates maximal cliques by repeatedly adding vertices to partial solutions, where each vertex is chosen according to the probability that depends on the level of pheromone. The pheromone deposition is proportional to the quality of previously constructed cliques. The performance of the algorithm was compared to that of [137] and showed that on average Ant-Clique is capable of reaching better solutions on the majority of tested instances. Youseff and Elliman [184] introduced an algorithm that combines the capabilities of ant colony optimization and local search techniques with prohibition rules. The algorithm was compared to the genetic local search of [137] and has shown to be somewhat competitive. Incorporation of local search techniques into the ant colony algorithm has also been addressed in [165] and has shown to improve the solution process.

One of the first ant colony algorithms for the vertex coloring problem was designed by Costa et al. [43]. Their algorithm, called ANTCOL, considers each ant as a constructive heuristic derived from DSATUR [23] and RLF [125]. Even though the algorithm was not superior of the best graph coloring heuristics of that time, its performance was still promising enough to encourage more research in this direction. Dowsland and Thompson further applied the same algorithm for examination scheduling [53] and later enhanced it by strengthening the construction phase and incorporating it with a tabu search improvement phase [54].

Hertz and Zuffrey [100] argued that an algorithm, where each ant has only a minor role, is still competitive with the other ant algorithms. Namely, in their proposed algorithm, each single ant does not build an entire solution but only colors a single vertex. A similar idea was explored by Bui et al. [27]. In their algorithm, instead of coloring the entire graph, each ant colors only a portion of the graph. The performance of the algorithm seemed to be rather encouraging. Plumetta et al. [155] proposed an ant local search algorithm, where each ant is considered as a local search, rather than a constructive heuristic. The algorithm showed to be competitive with other evolutionary methods available at that time.

Recently, Zufferey [188] conducted an analysis of importance of different ant roles, ranging from insignificant ones, used to make a minor decision, to more crucial ones, by performing a refined local search. It was shown experimentally that the ant algorithms are more efficient when ants have strong roles like local search procedures.

Heuristics Based on Continuous Formulations

A remarkable result by Motzkin and Straus [144], which provides an elegant connection between maximum cliques and a certain quadratic program, has been extensively explored to devise efficient heuristics for the maximum clique problem. Some of the early methods are discussed in detail in [22].

Given a graph $G = (V, E)$, let A_G be the adjacency matrix of G. Consider the following quadratic problem:

$$\begin{aligned} \max\ & g(x) = x^T A_G x, \\ \text{s.t. } & e^T x = 1, \\ & x \in R^{|V|}, \end{aligned}$$

where e is a unit vector.

Motzkin and Straus [144] showed that the size of a maximum clique in G is equal to:

$$\omega(G) = \frac{1}{1 - g(x^*)},$$

where x^* is a global maximizer of the above problem.

One drawback associated with this result is that a solution vector x^* does not always correspond to a maximum clique C, hence, making it hard to extract the clique's vertices. Bomze [21] proposed a regularization of the Motzkin-Straus formulation by adding $\frac{1}{2} x^T I x$ to the objective, where I is the corresponding identity matrix. He has shown that every local maximum of the modified formulation corresponds to a maximal clique in the graph as follows: C is a maximal clique of G if and only if x^*, such that $x_i^* = 1/|C|$ if $x_i \in C$ and $x_i^* = 0$ otherwise, is a local maximum of the regularized Motzkin-Straus formulation.

One of the early approaches based on the Motzkin-Straus result was an iterative clique retrieval procedure [149], which turned out to be of very high computational cost and was only able to solve instances on less than 75 vertices. Gibbons et al. [79] proposed to modify the Motzkin-Straus formulation so that it becomes a problem of optimizing a quadratic function over a sphere. Even though such problem is polynomially solvable, it, however, yields approximate solutions that need to be rounded. Similar approaches, with a few advances, can be found in [30,31]. Simple heuristics based on formulations of the maximum independent set problem as maximization of polynomial functions over a unit hypercube were studied in [3].

Gruzdeva [89] proposed to add a non-convex quadratic constraint represented by the difference of two convex functions (DC contraints) [105] to a continuous formulation of the maximum clique problem. The author showed that the proposed approach is competitive with other methods based on continuous formulations.

Massaro et al. [138] transformed the Motzkin-Straus formulation of the maximum clique problem into its corresponding linear complementary problem (LCP) [45]. To deal with the inherent degeneracy of the derived LCP, the authors designed a variation of a classical Lemke's method [126] with an effective "look-ahead" pivot rule.

Recently, Butenko et al. [34] proposed variable objective search, a metaheuristic framework that performs a local search with respect to different alternative formulations of the same combinatorial optimization problems. To test their method, authors considered the maximum independent set problem and its two equivalent non-convex programs discussed in [3].

Another notable result – the "sandwich theorem" [122] – has inspired appearance of several heuristic algorithms for both the maximum clique (independent set) [28, 55] and vertex coloring problems [56, 84, 117]. For a given graph G, the sandwich theorem states that the following relationship holds:

$$\omega(G) \leq \theta(\bar{G}) \leq \chi(G),$$

where $\theta(\bar{G})$ is the Lovász theta, which can be computed in polynomial time [129].

In [28], the authors study rank-one and rank-two formulations of the semidefinite programming (SDP) formulation of the Lovász theta number. Based on the obtained further continuous formulations, they designed heuristics for the maximum independent set problem. As for the vertex coloring problem, the proposed solution algorithms also used a semidefinite programming (SDP) formulation of the Lovász theta and were capable of obtaining near-optimal solutions for problems of medium sizes [56].

Other Heuristics

Many other interesting heuristic approaches have been devised to tackle the maximum clique and the vertex coloring problems. Balas and Niehaus [10] developed a heuristic, called CLIQMERGE, that generates large cliques by repeatedly finding a maximum clique in the subgraph induced by the union of two cliques. Such procedure is performed by finding the bipartite matching of the complement subgraph. A heuristic for the maximum independent set problem based on the operation of *edge projection*, which is a specialization of Lovász and Plummer's clique projection [130], has been designed in [135]. Goldberg and Rivenburgh [83] used a *restricted backtracking* for detecting cliques in graphs. DNA computing was applied to the maximum clique problem in [147, 185]. Neural network approach has been very popular in tackling the maximum clique problem since the mid-1980s. A detailed discussion on neural network-based methods applied to the maximum clique problem developed before 1999 is presented in [22]. The interested readers are also referred to [86, 106, 107, 110].

For the vertex coloring problem, the neural network approach has also been applied in [16, 17, 77, 108, 154, 168, 169]. An algorithm which inherits ideas from quantum mechanics was proposed in [171].

Computational Results

In this section we summarize best computational results achieved by heuristic algorithms on selected hard benchmark instances for the problems of interest. We disregard running times and focus on the quality of the reported solutions. More

specifically, we only report the best found solutions for the instances with no known optima and list the methods that were able to attain these solutions.

The results for the maximum clique instances are presented in Table 1. First three columns contain information about the graph instance (name, number of vertices $|V|$, and edges $|E|$). In the next columns, we report the size of the largest detected clique (ω_{lb}) and list references of the methods that were able to attain such solution. Note that the first two graph instances in Table 1 are from the Second DIMACS Implementation Challenge [49], the next three instances are from the Tenth DIMACS Implementation Challenge [50], and the rest of them are from the so-called CODE family [164], which is a collection of challenging graph instances arising from the coding theory.

The computational results related to the coloring problem are contained in Table 2. Here, the first three columns are the same as in Table 1. The fourth and fifth columns contain the size of the lower (χ_{lb}) and upper (χ_{ub}) bounds on the chromatic number. The majority of the results related to the lower bound size is by [98]. The last column in Table 2 lists the references to the algorithms which have achieved χ_{ub}. The first 27 instances presented in this table are from the Second DIMACS Implementation Challenge [49]; the next 12 are from Stanford Large Network Dataset Collection [127], referred to as the SNAP; and the last 8 are from the Tenth DIMACS Implementation Challenge [50].

More detailed discussions on computational performance of various heuristics can be found in some of the recent surveys [73, 132, 182].

Table 1 Approximate solutions to the maximum clique problem

Instance	$\|V\|$	$\|E\|$	ω_{lb}	ω_{lb} obtained by
Keller6	3,361	4,619,898	59	[5, 14, 15, 90, 95, 112, 158–160, 180, 183]
Hamming10-4	1,024	434,176	40	[14, 15, 90, 95, 112, 158, 159, 180, 183]
c500.9	500	112,332	57	[14, 15, 90, 95, 112, 158–160, 180, 183]
c1000.9	1,000	450,079	68	[5, 14, 15, 90, 95, 112, 158–160, 180, 183]
c2000.9	2,000	1,799,532	80	[15, 112, 180, 183]
1dc.1024	1,024	24,063	94	[5, 33, 112]
1dc.2048	2,048	58,367	172	[5, 33, 112]
1et.1024	1,024	9,600	171	[5, 33, 112]
1et.2048	2,048	22,528	316	[5, 33, 112]
1tc.1024	1,024	7,936	196	[5, 33, 112]
1tc.2048	2,048	18,944	352	[5, 33, 112]
1zc.1024	1,024	33,280	112	[5, 33, 112]
1zc.2048	2,048	78,848	198	[5, 33, 112]
1zc.4096	4,096	184,320	379	[5, 33, 112]
2dc.1024	1,024	169,162	16	[5, 33, 112]
2dc.2048	2,048	504,451	24	[5, 33, 112]

Table 2 Approximate solutions to the vertex coloring problem

Instance	$\|V\|$	$\|E\|$	χ_{lb}	χ_{ub}	χ_{ub} obtained by
DSJC250.5	250	15,668	26	28	[40, 69, 70, 72, 84, 131, 133, 142, 157, 171]
DSJC500.1	500	12,458	9	12	[18, 40, 52, 69, 72, 84, 101, 131, 133, 142, 155–157, 171]
DSJC500.5	500	62,624	43	48	[18, 52, 70, 72, 101, 131, 133, 142, 155–157, 170, 171]
DSJC500.9	500	1,124,367	123	126	[40, 40, 52, 72, 101, 131, 155–157, 170, 171]
DSJC1000.1	1,000	49,629	10	20	[18, 52, 70, 72, 96, 101, 131, 133, 155–157, 170, 171, 179]
DSJC1000.5	1,000	249,826	73	83	[52, 70, 96, 131, 133, 142, 156, 170, 171, 179]
DSJC1000.9	1,000	449,449	216	222	[96, 170, 171, 179]
r1000.1c	1000	485090	96	98	[18, 69, 96, 131, 133, 142, 156, 157, 171]
latin_square_10	900	307,350	90	97	[170]
1-Insertions_5	202	1,227	4	6	[72, 134]
1-Insertions_6	607	6,337	4	7	[72, 134]
2-Insertions_4	149	541	4	5	[40, 72, 134]
2-Insertions_5	597	3,936	3	6	[72, 134]
3-Insertions_4	281	1,046	3	4	[40]
3-Insertions_5	1,406	9,695	3	6	[40, 134]
4-Insertions_4	475	1,795	3	5	[72, 134]
4-FullIns_5	4,146	77,305	7	9	[40, 134]
5-FullIns_4	1,085	11,395	8	9	[134]
wap01	2,368	110,871	41	42	[40, 96]
wap02	2,464	111,742	40	41	[96]
wap03	4,730	286,722	40	44	[96]
wap04	5,231	294,902	40	42	[96]
wap07	1,809	103,368	40	41	[96]
wap08	1,870	104,176	40	42	[96, 134]
c2000.5	2,000	999,836	99	145	[96]
c2000.9	2,000	1,799,532	80	408	[96]
c4000.5	4,000	4,000,268	107	259	[96]
Wiki-Vote	7,115	100,762	19	24	[172]
p2p-Gnutella04	10,876	39,994	4	6	[172]
p2p-Gnutella25	22,687	54,705	4	6	[172]
p2p-Gnutella24	26,518	65,369	4	6	[172]
Cit-HepTh	27,770	352,285	23	25	[172]
Cit-HepPh	34,546	420,877	19	21	[172]
p2p-Gnutella30	36,682	88,328	4	6	[172]
p2p-Gnutella31	62,586	147,892	4	6	[172]
soc-Epinions1	75,879	405,740	25	30	[172]
Email-EuAll	265,214	364,481	18	20	[172]
WikiTalk	2,394,385	4,659,565	31	51	[172]
cit-Patents	3,774,768	16,518,947	11	12	[172]

(*continued*)

Table 2 (continued)

Instance	$\|V\|$	$\|E\|$	χ_{lb}	χ_{ub}	χ_{ub} obtained by
kron_g500-simple-logn16	65,536	2,456,071	136	155	[172]
G_n_pin_pout	100,000	501,198	4	8	[172]
smallworld	100,000	499,998	6	8	[172]
wave	156,317	1,059,331	6	9	[172]
audikw1	943,695	38,354,076	36	44	[172]
ldoor	952,203	22,785,136	21	35	[172]
333SP	3,712,815	11,108,633	4	5	[172]
cage15	5,154,859	47,022,346	6	13	[172]

Conclusion

In this chapter, we discussed various heuristics for approximating the maximum clique and the vertex coloring problems that were developed in the past 20–30 years. The local search-based metaheuristics proved to be more effective than the population-based ones. Nevertheless, the latter ones still have gained a lot of attention among the researchers. Indeed, when combined with effective local search techniques, they can actually yield competitive results; moreover, they can be easily parallelized. Heuristics based on continuous optimization represent an interesting and promising direction for future research.

Cross-References

- GRASP
- Tabu Search
- Variable Neighborhood Search

Acknowledgments We would like to thank the anonymous reviewers and the book editors, whose suggestions helped to improve the manuscript. Partial support of NSF grant CMMI-1538493 is also gratefully acknowledged.

References

1. Aarts E, Korst J (1988) Simulated annealing and Boltzmann machines. Wiley, Chichester
2. Abello J, Pardalos PM, Resende MGC (1999) On maximum clique problems in very large graphs. In: Abello J, Vitter J (eds) External memory algorithms and visualization. American Mathematical Society, Boston, pp 119–130
3. Abello J, Butenko S, Pardalos PM, Resende MGC (2001) Finding independent sets in a graph using continuous multivariable polynomial formulations. J Glob Optim 21(2):111–137
4. Aggarwal CC, Orlin JB, Tai RP (1997) Optimized crossover for the independent set problem. Oper Res 45(2):226–234

5. Andrade DV, Resende MGC, Werneck RF (2012) Fast local search for the maximum independent set problem. J Heuristics 18(4):525–547
6. Arora S, Safra S (1998) Probabilistic checking of proofs: a new characterization of NP. J ACM (JACM) 45(1):70–122
7. Arora S, Lund C, Motwani R, Sudan M, Szegedy M (1998) Proof verification and the hardness of approximation problems. J ACM (JACM) 45(3):501–555
8. Avanthay C, Hertz A, Zufferey N (2003) A variable neighborhood search for graph coloring. Eur J Oper Res 151(2):379–388
9. Back T, Khuri S (1994) An evolutionary heuristic for the maximum independent set problem. In: Proceedings of the first IEEE conference on evolutionary computation, pp 531–535
10. Balas E, Niehaus W (1996) Finding large cliques in arbitrary graphs by bipartite matching. DIMACS Ser Discrete Math Theor Comput Sci 26:29–52
11. Balas E, Niehaus W (1998) Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems. J Heuristics 4(2):107–122
12. Balasundaram B, Butenko S (2006) Graph domination, coloring and cliques in telecommunications. In: Handbook of optimization in telecommunications. Springer, Berlin, pp 865–890
13. Battiti R, Mascia F (2010) Reactive and dynamic local search for max-clique: engineering effective building blocks. Comput Oper Res 37(3):534–542
14. Battiti R, Protasi M (2001) Reactive local search for the maximum clique problem. Algorithmica 29(4):610–637
15. Benlic U, Hao JK (2013) Breakout local search for maximum clique problems. Comput Oper Res 40(1):192–206
16. Berger MO (1994) k-coloring vertices using a neural network with convergence to valid solutions. In: Proceedings of IEEE international conference on neural networks, vol 7, pp 4514–4517
17. Blas AD, Jagota A, Hughey R (2002) Energy function-based approaches to graph coloring. IEEE Trans Neural Netw 13(1):81–91
18. Blöchliger I, Zufferey N (2008) A graph coloring heuristic using partial solutions and a reactive tabu scheme. Comput Oper Res 35(3):960–975
19. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv (CSUR) 35(3):268–308
20. Bollobás B, Thomason A (1985) Random graphs of small order. North-Holland Math Stud 118:47–97
21. Bomze IM (1997) Evolution towards the maximum clique. J Glob Optim 10:143–164
22. Bomze IM, Budinich M, Pardalos PM, Pelillo M (1999) The maximum clique problem. In: Handbook of combinatorial optimization. Springer, Boston, pp 1–74
23. Brélaz D (1979) New methods to color the vertices of a graph. Commun ACM 22(4):251–256
24. Brooks RL (1941) On colouring the nodes of a network. In: Mathematical proceedings of the Cambridge philosophical society. Cambridge University Press, vol 37, pp 194–197
25. Brunato M, Battiti R (2011) R-EVO: a reactive evolutionary algorithm for the maximum clique problem. IEEE Trans Evol Comput 15(6):770–782
26. Bui TN, Eppley PH (1995) A hybrid genetic algorithm for the maximum clique problem. In: Proceedings of the 6th international conference on genetic algorithms. Morgan Kaufmann Publishers Inc., pp 478–484
27. Bui TN, Nguyen TH, Patel CM, Phan KAT (2008) An ant-based algorithm for coloring graphs. Discrete Appl Math 156(2):190–200
28. Burer S, Monteiro RD, Zhang Y (2002) Maximum stable set formulations and heuristics based on continuous optimization. Math Program 94(1):137–166
29. Burke E, Kendall G, Newall J, Hart E, Ross P, Schulenburg S (2003) Hyper-heuristics: an emerging direction in modern search technology. Int Ser Oper Res Manag Sci 57:457–474
30. Busygin S (2006) A new trust region technique for the maximum weight clique problem. Discrete Appl Math 154(15):2080–2096
31. Busygin S, Butenko S, Pardalos PM (2002) A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere. J Comb Optim 6(3):287–297

32. Butenko S, Wilhelm WE (2006) Clique-detection models in computational biochemistry and genomics. Eur J Oper Res 173(1):1–17
33. Butenko S, Pardalos PM, Sergienko IV, Shylo V, Stetsyuk P (2009) Estimating the size of correcting codes using extremal graph problems. In: Pearce C, Hunt E (eds) Optimization: structure and applications. Springer, New York, pp 227–243
34. Butenko S, Yezerska O, Balasundaram B (2013) Variable objective search. J Heuristics 19(4):697–709
35. Caprara A, Kroon L, Monaci M, Peeters M, Toth P (2007) Passenger railway optimization. Handb Oper Res Manag Sci 14:129–187
36. Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. Oper Res Lett 9(6):375–382
37. Carter R, Park K (1993) How good are genetic algorithms at finding large cliques: an experimental study. Technical report, Computer Science Department, Boston University
38. Chaitin GJ, Auslander MA, Chandra AK, Cocke J, Hopkins ME, Markstein PW (1981) Register allocation via coloring. Comput Lang 6(1):47–57
39. Chams M, Hertz A, de Werra D (1987) Some experiments with simulated annealing for coloring graphs. Eur J Oper Res 32(2):260–266
40. Chiarandini M, Stützle T et al (2002) An application of iterated local search to graph coloring problem. In: Proceedings of the computational symposium on graph coloring and its generalizations, pp 112–125
41. Chiarandini M, Dumitrescu I, Stützle T (2007) Stochastic local search algorithms for the graph colouring problem. In: Handbook of approximation algorithms and metaheuristics. Chapman & Hall/CRC, Boca Raton, pp 63-1
42. Chow FC, Hennessy JL (1990) The priority-based coloring approach to register allocation. ACM Trans Program Lang Syst (TOPLAS) 12(4):501–536
43. Costa D, Hertz A (1997) Ants can colour graphs. J Oper Res Soc 48(3):295–305
44. Costa D, Hertz A, Dubuis C (1995) Embedding a sequential procedure within an evolutionary algorithm for coloring problems in graphs. J Heuristics 1(1):105–128
45. Cottle RW, Pang JS, Stone RE (1992) The linear complementarity problem. SIAM, Philadelphia
46. Culberson JC (1992) Iterated greedy graph coloring and the difficulty landscape. Technical report. TK 92-07, Department of Computing Science, University of Alberta
47. Culberson JC, Luo F (1996) Exploring the k-colorable landscape with iterated greedy. In: Johnson DS, Trick MA (eds) Cliques, coloring, and satisfiability: second DIMACS implementation challenge. American Mathematical Society, Providence, pp 245–284
48. Davis L (1991) Order-based genetic algorithms and the graph coloring problem. In: Handbook of genetic algorithms. Van Nostrand Reinhold, New York, pp 72–90
49. DIMACS (1993) NP hard problems: maximum clique, graph coloring, and satisfiability. The second DIMACS implementation challenge. http://dimacs.rutgers.edu/Challenges/. Accessed 10 Jan 2018
50. DIMACS (2012) Algorithm implementation challenge: graph partitioning and graph clustering. The tenth DIMACS implementation challenge. http://dimacs.rutgers.edu/Challenges/. Accessed 10 Jan 2018
51. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B Cybern 26(1):29–41
52. Dorne R, Hao JK (1998) A new genetic local search algorithm for graph coloring. In: Parallel problem solving from nature. Springer, Berlin/Heidelberg, pp 745–754
53. Dowsland KA, Thompson JM (2005) Ant colony optimization for the examination scheduling problem. J Oper Res Soc 56(4):426–438
54. Dowsland KA, Thompson JM (2008) An improved ant colony optimisation heuristic for graph colouring. Discrete Appl Math 156(3):313–324
55. Dukanovic I, Rendl F (2007) Semidefinite programming relaxations for graph coloring and maximal clique problems. Math Program 109(2–3):345–365

56. Dukanovic I, Rendl F (2008) A semidefinite programming-based heuristic for graph coloring. Discrete Appl Math 156(2):180–189
57. Eiben ÁE, Van Der Hauw JK, van Hemert JI (1998) Graph coloring with adaptive evolutionary algorithms. J Heuristics 4(1):25–46
58. Erdös P (1970) On the graph theorem of Turán. Mat Lapok 21(249–251):10
59. Feige U, Kilian J (1996) Zero knowledge and the chromatic number. In: Proceedings of eleventh annual IEEE conference on computational complexity, pp 278–287
60. Feige U, Kilian J (1998) Zero knowledge and the chromatic number. J Comput Syst Sci 57:187–199
61. Fenet S, Solnon C (2003) Searching for maximum cliques with ant colony optimization. In: Applications of evolutionary computing. Springer, Berlin, pp 236–245
62. Feo TA, Resende MGC (1989) A probabilistic heuristic for a computationally difficult set covering problem. Oper Res Lett 8:67–71
63. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Glob Optim 6:109–133
64. Feo TA, Resende MGC, Smith SH (1994) A greedy randomized adaptive search procedure for maximum independent set. Oper Res 42(5):860–878
65. Ferland J, Fleurent C (1996) Object-oriented implementation of heuristic search methods for graph coloring, maximum clique and satisfiability. In: Johnson DS, Trick MA (eds) Cliques, coloring, and satisfiability: second DIMACS implementation challenge. American Mathematical Society, Providence, pp 619–652
66. Fleurent C, Ferland JA (1996) Genetic and hybrid algorithms for graph coloring. Ann Oper Res 63(3):437–461
67. Foster JA, Soule T (1995) Using genetic algorithms to find maximum cliques. Technical report. LAL95-12, Department of Computer Science, University of Idaho
68. Friden C, Hertz A, de Werra D (1989) STABULUS: a technique for finding stable sets in large graphs with tabu search. Computing 42:35–44
69. Funabiki N, Higashino T (2000) A minimal-state processing search algorithm for graph coloring problems. IEICE Trans Fundam Electron Commun Comput Sci 83(7):1420–1430
70. Galinier P, Hao JK (1999) Hybrid evolutionary algorithms for graph coloring. J Comb Optim 3(4):379–397
71. Galinier P, Hertz A (2006) A survey of local search methods for graph coloring. Comput Oper Res 33(9):2547–2562
72. Galinier P, Hertz A, Zufferey N (2008) An adaptive memory algorithm for the k-coloring problem. Discrete Appl Math 156(2):267–279
73. Galinier P, Hamiez JP, Hao JK, Porumbel D (2013) Recent advances in graph vertex coloring. In: Handbook of optimization. Springer, Berlin/Heidelberg, pp 505–528
74. Gamst A (1986) Some lower bounds for a class of frequency assignment problems. IEEE Trans Veh Technol 35(1):8–14
75. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company, New York
76. Garey MR, Johnson DS, So H (1976) An application of graph coloring to printed circuit testing. IEEE Trans Circuits Syst 23(10):591–599
77. Gassen DW, Carothers JD (1993) Graph color minimization using neural networks. In: Proceedings of IEEE international joint conference on neural networks, vol 2, pp 1541–1544
78. Gendreau M, Soriano P, Salvail L (1993) Solving the maximum clique problem using a tabu search approach. Ann Oper Res 41(4):385–403
79. Gibbons LE, Hearn DW, Pardalos PM (1996) A continuous based heuristic for the maximum clique problem. In: Johnson DS, Trick MA (eds) Cliques, coloring, and satisfiability: second DIMACS implementation challenge. American Mathematical Society, Providence, pp 103–124
80. Glass CA, Prügel-Bennett A (2003) Genetic algorithm for graph coloring: exploration of Galinier and Hao's algorithm. J Comb Optim 7(3):229–236

81. Glover F (1989) Tabu search. Part I. ORSA J Comput 1(3):190–206
82. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading
83. Goldberg MK, Rivenburgh RD (1996) Constructing cliques using restricted backtracking. In: Johnson DS, Trick MA (eds) Cliques, coloring, and satisfiability: second DIMACS implementation challenge. American Mathematical Society, Providence, pp 285–307
84. Govorčin J, Gvozdenović N, Povh J (2013) New heuristics for the vertex coloring problem based on semidefinite programming. Cent Eur J Oper Res 21(1):13–25
85. Grable DA, Panconesi A (1998) Fast distributed algorithms for Brooks-Vizing colourings. In: Proceedings of the ninth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, pp 473–480
86. Grossman T (1996) Applying the inn model to the maximum clique problem. In: Johnson DS, Trick MA (eds) Cliques, coloring, and satisfiability: second DIMACS implementation challenge. American Mathematical Society, Providence, pp 125–146
87. Grosso A, Locatelli M, Della Croce F (2004) Combining swaps and node weights in an adaptive greedy approach for the maximum clique problem. J Heuristics 10(2):135–152
88. Grosso A, Locatelli M, Pullan W (2008) Simple ingredients leading to very efficient heuristics for the maximum clique problem. J Heuristics 14(6):587–612
89. Gruzdeva TV (2013) On a continuous approach for the maximum weighted clique problem. J Glob Optim 56(3):971–981
90. Guturu P, Dantu R (2008) An impatient evolutionary algorithm with probabilistic tabu search for unified solution of some np-hard problems in graph and set theory via clique finding. IEEE Trans Syst Man Cybern B Cybern 38(3):645–666
91. Hajnal P, Szemerédi E (1990) Brooks coloring in parallel. SIAM J Discret Math 3(1):74–80
92. Hamiez JP, Hao JK (2002) Scatter search for graph coloring. In: Artificial evolution. Springer, Berlin/Heidelberg pp 168–179
93. Hansen P, Jaumard B (1990) Algorithms for the maximum satisfiability problem. Computing 44(4):279–303
94. Hansen P, Mladenović N (1999) An introduction to variable neighborhood search. In: Voss S et al (eds) Meta-heuristics: advances and trends in local search paradigms for optimization. Springer, Boston, pp 433–458
95. Hansen P, Mladenović N, Urošević D (2004) Variable neighborhood search for the maximum clique. Discret Appl Math 145(1):117–125
96. Hao JK, Wu Q (2012) Improving the extraction and expansion method for large graph coloring. Discret Appl Math 160(16):2397–2407
97. Håstad J (1999) Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math 182:105–142
98. Held S, Cook W, Sewell EC (2012) Maximum-weight stable sets and safe lower bounds for graph coloring. Math Program Comput 4(4):363–381
99. Hertz A, de Werra D (1987) Using tabu search techniques for graph coloring. Computing 39(4):345–351
100. Hertz A, Zufferey N (2006) A new ant algorithm for graph coloring. In: Workshop on nature inspired cooperative strategies for optimization, NICSO, pp 51–60
101. Hertz A, Plumettaz M, Zufferey N (2008) Variable space search for graph coloring. Discret Appl Math 156(13):2551–2560
102. Hifi M (1997) A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems. J Oper Res Soc 48(6):612–622
103. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge
104. Homer S, Peinado M (1996) Experiments with polynomial-time clique approximation algorithms on very large graphs. DIMACS Ser Discret Math Theor Comput Sci 26: 147–168
105. Horst R, Thoai NV (1999) Dc programming: overview. J Optim Theory Appl 103(1):1–43
106. Jagota A (1992) Efficiently approximating MAX-CLIQUE in a Hopfield-style network. In: International joint conference on neural networks, vol 2, pp 248–253

107. Jagota A (1995) Approximating maximum clique with a Hopfield network. IEEE Trans Neural Netw 6(3):724–735
108. Jagota A (1996) An adaptive, multiple restarts neural network algorithm for graph coloring. Eur J Oper Res 93(2):257–270
109. Jagota A, Sanchis LA (2001) Adaptive, restart, randomized greedy heuristics for maximum clique. J Heuristics 7(6):565–585
110. Jagota A, Sanchis L, Ganesan R (1996) Approximately solving maximum clique using neural networks and related heuristics. DIMACS Ser Discret Math Theor Comput Sci 26: 169–204
111. Jerrum M (1992) Large cliques elude the metropolis process. Random Struct Algorithm 3(4):347–359
112. Jin Y, Hao JK (2015) General swap-based multiple neighborhood tabu search for the maximum independent set problem. Eng Appl Artif Intell 37:20–33
113. Johnson DS, Trick MA (eds) (1996) Cliques, coloring, and satisfiability: second DIMACS implementation challenge. American Mathematical Society, Providence
114. Johnson DS, Aragon CR, McGeoch LA, Schevon C (1991) Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. Oper Res 39(3):378–406
115. Kahruman-Anderoglu S, Buchanan A, Butenko S, Prokopyev O (2016) On provably best construction heuristics for hard combinatorial optimization problems. Networks 67:238–245. https://doi.org/10.1002/net.21620
116. Karchmer M, Naor J (1988) A fast parallel algorithm to color a graph with Δ colors. J Algorithm 9(1):83–91
117. Karger D, Motwani R, Sudan M (1998) Approximate graph coloring by semidefinite programming. J ACM (JACM) 45(2):246–265
118. Karloff HJ (1989) An NC algorithm for Brooks' theorem. Theor Comput Sci 68(1):89–103
119. Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of computer computations. Plenum, New York, pp 85–103
120. Katayama K, Hamamoto A, Narihisa H (2005) An effective local search for the maximum clique problem. Inf Process Lett 95(5):503–511
121. Kirkpatrick S, Vecchi M et al (1983) Optimization by simulated annealing. Science 220(4598):671–680
122. Knuth DE (1994) The sandwich theorem. Electron J Comb 1:1–48. http://www.combinatorics.org/Volume_1/Abstracts/v1i1a1.html. Accessed 10 Jan 2018
123. Kopf R, Ruhe G (1987) A computational study of the weighted independent set problem for general graphs. Found Control Eng 12(4):167–180
124. Laguna M, Martí R (2001) A grasp for coloring sparse graphs. Computat Optim Appl 19(2):165–178
125. Leighton FT (1979) A graph coloring algorithm for large scheduling problems. J Res Natl Bur Stand 84(6):489–506
126. Lemke CE (1965) Bimatrix equilibrium points and mathematical programming. Manag Sci 11(7):681–689
127. Leskovec J, Krevl A (2014) SNAP datasets: Stanford large network dataset collection. http://snap.stanford.edu/data. Accessed 10 Jan 2018
128. Lovász L (1975) Three short proofs in graph theory. J Comb Theory Ser B 19(3):269–271
129. Lovász L (1979) On the shannon capacity of a graph. IEEE Trans Inf Theory 25(1):1–7
130. Lovász L, Plummer MD (2009) Matching theory. American Mathematical Society, Providence
131. Lü Z, Hao JK (2010) A memetic algorithm for graph coloring. Eur J Oper Res 203(1): 241–250
132. Malaguti E, Toth P (2010) A survey on vertex coloring problems. Int Trans Oper Res 17(1): 1–34
133. Malaguti E, Monaci M, Toth P (2008) A metaheuristic approach for the vertex coloring problem. INFORMS J Comput 20(2):302–316

134. Malaguti E, Monaci M, Toth P (2011) An exact approach for the vertex coloring problem. Discret Optim 8(2):174–190
135. Mannino C, Sassano A (1996) Edge projection and the maximum cardinality stable set problem. DIMACS Ser Discret Math Theor Comput Sci 26:205–219
136. Marchiori E (1998) A simple heuristic based genetic algorithm for the maximum clique problem. In: Proceedings of the ACM symposium on applied computing, vol 27, pp 366–373
137. Marchiori E (2002) Genetic, iterated and multistart local search for the maximum clique problem. In: Applications of evolutionary computing. Springer, Berlin/Heidelberg, pp 112–121
138. Massaro A, Pelillo M, Bomze IM (2002) A complementary pivoting approach to the maximum weight clique problem. SIAM J Optim 12(4):928–948
139. Matula DW, Marble G, Isaacson JD (1972) Graph coloring algorithms. In: Read R (ed) Graph theory and computing. Academic, New York, pp 109–122
140. Mehta NK (1981) The application of a graph coloring method to an examination scheduling problem. Interfaces 11(5):57–65
141. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100
142. Morgenstern C (1996) Distributed coloration neighborhood search. Discret Math Theor Comput Sci 26:335–358
143. Morgenstern CA, Shapiro HD (1986) Chromatic number approximation using simulated annealing. Technical report, Department of Computer Science, University of New Mexico
144. Motzkin TS, Straus EG (1965) Maxima for graphs and a new proof of a theorem of turán. Canad J Math 17(4):533–540
145. Mumford CL (2006) New order-based crossovers for the graph coloring problem. In: Parallel problem solving from nature. Springer, Berlin, pp 880–889
146. Murthy AS, Parthasarathy G, Sastry V (1994) Clique finding – a genetic approach. In: Proceedings of the first IEEE conference on evolutionary computation, pp 18–21
147. Ouyang Q, Kaplan PD, Liu S, Libchaber A (1997) Dna solution of the maximal clique problem. Science 278(5337):446–449
148. Paquete L, Stützle T (2002) An experimental investigation of iterated local search for coloring graphs. In: Applications of evolutionary computing. Springer, Berlin/Heidelberg, pp 122–131
149. Pardalos PM, Phillips A (1990) A global optimization approach for solving the maximum clique problem. Int J Comput Math 33(3–4):209–216
150. Pardalos PM, Xue J (1994) The maximum clique problem. J Glob Optim 4(3):301–328
151. Pardalos PM, Mavridou T, Xue J (1999) The graph coloring problem: a bibliographic survey. In: Handbook of combinatorial optimization. Springer, Boston, pp 1077–1141
152. Park K, Carter B (1995) On the effectiveness of genetic search in combinatorial optimization. In: Proceedings of the ACM symposium on applied computing, pp 329–336
153. Pattillo J, Butenko S (2011) Clique, independent set, and graph coloring. In: Encyclopedia of operations research and management science. Wiley, Hoboken, pp 3150–3163
154. Philipsen W, Stok L (1991) Graph coloring using neural networks. In: IEEE international symposium on circuits and systems, pp 1597–1600
155. Plumettaz M, Schindl D, Zufferey N (2010) Ant local search and its efficient adaptation to graph colouring. J Oper Res Soc 61(5):819–826
156. Porumbel DC, Hao JK, Kuntz P (2010) An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. Comput Oper Res 37(10):1822–1832
157. Porumbel DC, Hao JK, Kuntz P (2010) A search space "cartography" for guiding graph coloring heuristics. Comput Oper Res 37(4):769–778
158. Pullan W (2006) Phased local search for the maximum clique problem. J Comb Optim 12(3):303–323
159. Pullan W, Hoos HH (2006) Dynamic local search for the maximum clique problem. J Artif Intell Res 25:159–185

160. Pullan W, Mascia F, Brunato M (2011) Cooperating local search for the maximum clique problem. J Heuristics 17(2):181–199
161. Resende MGC, Feo TA, Smith SH (1998) Algorithm 787: fortran subroutines for approximate solution of maximum independent set problems using grasp. ACM Trans Math Softw (TOMS) 24(4):386–394
162. Singh A, Gupta AK (2006) A hybrid heuristic for the maximum clique problem. J Heuristics 12(1–2):5–22
163. Skulrattanakulchai S (2006) Δ-list vertex coloring in linear time. Inf Process Lett 98(3): 101–106
164. Sloane N (2000) Challenge problems: independent sets in graphs. https://oeis.org/A265032/a265032.html. Accessed 10 Jan 2018
165. Solnon C, Fenet S (2006) A study of ACO capabilities for solving the maximum clique problem. J Heuristics 12(3):155–180
166. Soriano P, Gendreau M (1996) Diversification strategies in tabu search algorithms for the maximum clique problem. Ann Oper Res 63(2):189–207
167. Soriano P, Gendreau M (1996) Tabu search algorithms for the maximum clique problem. In: Johnson DS, Trick MA (eds) Cliques, coloring, and satisfiability: second DIMACS implementation challenge. American Mathematical Society, Providence, pp 221–242
168. Takefuji Y, Lee KC (1991) Artificial neural networks for four-coloring map problems and k-colorability problems. IEEE Trans Circuits Syst 38(3):326–333
169. Talaván PM, Yáñez J (2008) The graph coloring problem: a neuronal network approach. Eur J Oper Res 191(1):100–111
170. Titiloye O, Crispin A (2011) Graph coloring with a distributed hybrid quantum annealing algorithm. In: Agent and multi-agent systems: technologies and applications. Springer, Berlin/Heidelberg, pp 553–562
171. Titiloye O, Crispin A (2011) Quantum annealing of the graph coloring problem. Discret Optim 8(2):376–384
172. Verma A, Buchanan A, Butenko S (2015) Solving the maximum clique and vertex coloring problems on very large sparse networks. INFORMS J Comput 27(1):164–177
173. Welsh DJ, Powell MB (1967) An upper bound for the chromatic number of a graph and its application to timetabling problems. Comput J 10(1):85–86
174. de Werra D (1985) An introduction to timetabling. Eur J Oper Res 19(2):151–162
175. de Werra D (1990) Heuristics for graph coloring. In: Computational graph theory. Springer, Berlin, pp 191–208
176. de Werra D, Gay Y (1994) Chromatic scheduling and frequency assignment. Discret Appl Math 49(1):165–174
177. Woo TK, Su SY, Newman-Wolfe R (1991) Resource allocation in a dynamically partitionable bus network using a graph coloring algorithm. IEEE Trans Commun 39(12):1794–1801
178. Wood D (1969) A technique for colouring a graph applicable to large scale timetabling problems. Comput J 12(4):317–319
179. Wu Q, Hao JK (2012) Coloring large graphs based on independent set extraction. Comput Oper Res 39(2):283–290
180. Wu Q, Hao JK (2013) An adaptive multistart tabu search approach to solve the maximum clique problem. J Comb Optim 26(1):86–108
181. Wu Q, Hao JK (2013) An extraction and expansion approach for graph coloring. Asia Pac J Oper Res 30(05):1350018
182. Wu Q, Hao JK (2015) A review on algorithms for maximum clique problems. Eur J Oper Res 242(3):693–709
183. Wu Q, Hao JK, Glover F (2012) Multi-neighborhood tabu search for the maximum weight clique problem. Ann Oper Res 196:611–634
184. Youssef SM, Elliman DG (2004) Reactive prohibition-based ant colony optimization (rpaco): a new parallel architecture for constrained clique sub-graphs. In: 16th IEEE international conference on tools with artificial intelligence, pp 63–70

185. Zhang BT, Shin SY (1999) Code optimization for dna computing of maximal cliques. In: Advances in soft computing. Springer, Heidelberg, pp 588–599
186. Zhang Q, Sun J, Tsang E (2005) An evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Trans Evol Comput 9(2):192–200
187. Zuckerman D (2007) Linear degree extractors and the inapproximability of max clique and chromatic number. Theory Comput 3:103–128
188. Zufferey N (2012) Optimization by ant algorithms: possible roles for an individual ant. Optim Lett 6(5):963–973