symmetry

MDPI

Article

The Maximum Clique Problem and Integer Programming Models, Their Modifications, Complexity and Implementation

Milos Seda

Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Technicka 2896/2, 623 00 Brno, Czech Republic; seda@fme.vutbr.cz

Abstract: The maximum clique problem is a problem that takes many forms in optimization and related graph theory problems, and also has many applications. Because of its NP-completeness (nondeterministic polynomial time), the question arises of its solvability for larger instances. Instead of the traditional approaches based on the use of approximate or stochastic heuristic methods, we focus here on the use of integer programming models in the GAMS (General Algebraic Modelling System) environment, which is based on exact methods and sophisticated deterministic heuristics incorporated in it. We propose modifications of integer models, derive their time complexities and show their direct use in GAMS. GAMS makes it possible to find optimal solutions to the maximum clique problem for instances with hundreds of vertices and thousands of edges within minutes at most. For extremely large instances, good approximations of the optimum are given in a reasonable amount of time. A great advantage of this approach over all the mentioned algorithms is that even if GAMS does not find the best known solution within the chosen time limit, it displays its value at the end of the calculation as a reachable bound.

Keywords: clique; independent set; GAMS; NP-complete problem; integer programming

Citation: Seda, M. The Maximum Clique Problem and Integer Programming Models, Their Modifications, Complexity and Implementation. *Symmetry* **2023**, *15*, 1979. https://doi.org/10.3390/sym15111979

Academic Editor: Lorentz Jäntschi

Received: 29 September 2023
Revised: 24 October 2023
Accepted: 24 October 2023
Published: 26 October 2023

1. Introduction

Assuming that a graph G is simple, i.e., it does not have self-loops and multi edges, the *Maximum Clique Problem* (abbreviated to MCP) consists in finding the complete subgraph of G with the largest number of vertices, called the *maximum clique* in the graph [1–5].

The problem has many applications, e.g., in electrical engineering, chemistry, biology, medicine, image processing, and networks analysis. In [6], a bottom-up clustering algorithm based on recursive collapsing of small cliques in a graph, applied to VLSI (Very-Large-Scale Integration) design, is proposed. An application from the gas industry is described in [7]. In [8], a graph-theoretic clique finding a method for comparative modelling of protein structures is presented. Clusters of proteins using a clique-based approach are also found in [9]. A clique-based method for multimodal brain tumor segmentation that considers a brain tumor image as a graph segment was applied in [10]. In [11], synaptic networks in the brain are studied and it was discovered that they contain an abundance of cliques of neurons bound into cavities that guide the emergence of correlated activities. In [12], for cliques in images, the clique potential is calculated to be used in the image segmentation. Cliques as complete subgraphs are also identified in social networks [13–15] to find out groups of people who all know each other. It is clear that the members inside the group (i.e., inside the clique of the graph) have a symmetric relationship with each other and a potentially asymmetric relationship with respect to objects outside the group (clique) since, with some objects outside the group (clique), they may have a close relationship (connected by an edge in the graph) while having no relationship with others (being not connected by an edge in the graph). From another perspective, a weaker symmetry relationship is contained in the representation of the graph by the adjacency matrix considered below because, since the graph is undirected, the adjacency matrix is symmetric.

In addition to practical applications of maximum clique in the graph, we can distinguish between papers that focus theoretically on the properties of cliques and specific sub-problems and those that deal with the implementation of methods for solving the underlying problem and testing them on representative sets of benchmarks.

Theoretically oriented articles can be divided into three groups:

(1) *Literature concerning properties of cliques*

In [16], the subgraph isomorphism is reduced to the clique problem. In [17], the clique problem is reduced to related maximization problems by a procedure known from NP-completeness proofs and the possibilities of expressing approximation ratios are explored. In [18], the maximum edge clique partitioning problem and its complexity with respect to the clique number is explored. In [19], the relationship between the independence and clique numbers is investigated. Upper and lower bounds for the clique number are derived in [20]. Further approximations are presented in [21,22]. Computing bounds on the clique number of a graph is solved using a sequential elimination algorithm in [23]. An algorithm for clique covers is presented and the exact upper bound on the size of a minimal clique cover is derived in [24]. Finding maximum hereditary structures in graphs based on cliques is presented in [25]. In [26], the notion of distance between vertices, where two vertices are adjacent if they have exactly a given distance, is generalized to cliques in exact distance power graphs of a given maximum degree. In [27], the clique recognition problem (i.e., whether a given graph is a clique graph) is applied to split clique graphs with their complexity explored. The special quasi-clique problem is defined in [28], mathematical programming formulations are proposed and results obtained by the Xpress-IVE solver are presented. In [29], the relationship between the maximum number of edges in hypergraphs with fixed matching and the clique number are investigated. In [30], a clique graph merging strategy to reduce large structured positive semidefinite matrix constraints in semidefinite programs is proposed. In [31] and [32], clique separators are used in solving the Maximum Weight Stable Set Problem. In [33], clique constraints are applied in solving the Fairness-Oriented Crew Rostering Problem. The connection between polynomial optimization, maximum cliques and Turán densities is explored in [34].

(2) *Maximum clique problems and their definition and solution in specific representations*

In [35], the maximum weighted clique problem (see Section 2) is represented by matroids and solved by a greedy algorithm. In [36], clique search problems are decomposed into smaller instances based on vertex and edge colourings. Variations of clique transversal and clique independent sets on graphs are studied in [37,38].

(3) *Relationship of clique problems with other optimization problems*

The maximum clique constrained to visibility graphs of a simple polygon, computed using dynamic programming, is described in [39]. In [40], it is restricted to one-planar graphs, in [41] to dense graphs and in [42] to graphs without long cycles. Papers [43,44] deal with a *dominating clique*, i.e., a clique that is also a *dominating set* of a graph $G = (V, E)$, which is a subset V' of V such that any vertex of G is either in V' or has a neighbour in V'. An algorithm for minimum covering by cliques in claw-free perfect graphs is introduced in [45]. The paper [46] deals with the clique-critical graphs, i.e., graphs whose clique graph changes whenever a vertex is removed. The relationship to job shop scheduling is described in [47].

In [48], Karp proved that the decision version of the problem whether a complete subgraph with k vertices exists in a graph G belongs to the class of NP-complete problems. To solve large instances of NP-complete and NP-hard problems, stochastic heuristic methods are frequently used. They can be classified according to several aspects [49]: (i) single solution-based algorithms (Local Search, Simulated Annealing, Hill Climbing, tabu search) and population solution-based algorithms (e.g., Genetic Algorithm, bio-inspired algorithms), or (ii) with another point of view applied to their behaviour: evolution-based algorithms (Genetic Algorithm, Genetic Programming, Differential Evolution), swarm intelligence-based algorithms (e.g., Ant Colony Optimisation, Particle Swarm Optimi-

sation), physics-based algorithms (e.g., Simulated Annealing and Harmony Search) and human behaviour-based algorithms (e.g., teaching–learning-based optimization algorithm).

Some of these methods have also been applied to solve the maximum clique problem. Prohibition-based techniques for neighbours are also used in the Reactive Local Search, proposed in [50]. The paper [51] proposes a hybrid algorithm created by combining an exact algorithm with the tabu search. It also uses the relationship between the chromatic number and the maximum clique in the graph. If a graph G contains a clique with n vertices, then just to colour the clique itself it is necessary to use n colours. Thus, the chromatic number $\chi(G)$ must be greater than or equal to clique number $\omega(G)$, i.e., $\chi(G) \geq \omega(G)$ [52]. These two values are often identical. In [53], ten different neighbourhoods in the search were used. An incremental DNA algorithm is implemented in [54], but it also gives incorrect solutions. In [55], a heuristic approach for finding maximum clique using minimal independent set in a graph is described. The heuristic, described in [56], is based on generating all subgraphs and eliminating those that cannot be cliques. In [57], a heuristic decision-making method for local density of the vertices in the undirected graph is proposed, a density index function is constructed and a decision-making inference for finding maximum cliques is made. In [58], a vertex trust indicator for vertices is calculated at each iteration and vertices with a lower degree than the current clique number are eliminated. In the heuristic of [59], the maximum clique problem is expressed as a linearly constrained quadratic maximization program and a similar approach, implemented in MATLAB, is described in [60]. The optimization behaviour of randomized search heuristics on sparse semi-random graphs is theoretically investigated in [61]. The paper [62] is a summary of heuristics for maximal clique and independent sets.

However, in solving the maximum clique problem the use of stochastic heuristic methods has many disadvantages. The basic problem is that the operations of heuristic methods, such as neighbourhood in simulated annealing and tabu search, and crossover and mutation in the genetic algorithm, generate a large number of infeasible solutions that must be penalized, thus increasing computational complexity, and the results are not satisfactory [63].

For these reasons, solutions using mathematical models and exact methods are more common for the problem under study. Linear programming and mixed integer programming approaches are applicable and such models are presented in [3,64–66]. There are also approaches based on deterministic methods such as branch and bound [67–70]. The exact algorithms, expressed as codes in programming languages, are described in [44,71–73]. No mathematical models are given here and the proposed algorithms can be considered deterministic heuristics.

In this paper, we will focus on the use of integer programming models and their applicability in the GAMS (General Algebraic Modelling System) programming tool, which has already been successfully used in previous papers to solve NP-complete set covering problems [74], in finding the Steiner minimum tree in networks [75] and solving the Travelling Salesman Problem for instances up to 100 cities [76]. But it is always necessary to have a suitable mathematical model, which can be then implemented in the GAMS environment. For these reasons, the models used had to be modified. This is described in detail, including GAMS source codes, in Section 3, and computational results are summarised in Section 4.

Based on the literature reviewed, this paper's contributions could be seen as follows:

1. Unconventional is the use of the professional software tool GAMS in solving the studied maximum clique problem in the graph. Of the articles cited, only one mentions the use of sw Xpress-IVE [28], with nothing more detailed found here on the implementation. Here, on the other hand, the implementation was the motivation for modifying the previously known models for GAMS with all the necessary information about the code presented in the text.
2. The modification of the models is not an end in itself as the original models work with the complementary matrix. It is, however, no longer necessary in the modified models because its values are determined by absolute-value equations. This reduces

the space complexity of the algorithm. In this way, the modified models do not strictly follow the scheme of linear or mixed integer programming problems. However, since the absolute value function is among the built-in functions of GAMS, this does not limit the use of the MIP solver.

3. In the paper, we also deal with theoretical aspects—two mathematical theorems concerning the time complexity of the proposed model modifications have been formulated and proved. In the referenced papers presenting the calculations, on the other hand, this aspect is not considered.
4. The computational power of GAMS has been steadily increasing thanks to the collaboration of development teams from prestigious institutions, and now it is possible to obtain good-quality results (optimal or near-optimal) even for quite large problem instances within a reasonable time and on an simple laptop. This is because deterministic heuristics have already been included in the computational kernel.
5. A great advantage of GAMS over all the mentioned algorithms is that even if it does not find the best known solution within the chosen time limit, it displays its value at the end of the calculation as a reachable bound. To achieve this, it is enough to increase the computation time and/or use a more powerful computer. This is important for instances where the optimal solution (or best known solution) is not archived, as is the case with the well-known DIMACS benchmarks. For example, for a timeout of 100 s, benchmark gen400_p0.9_55 and the first model GAMS finds the current solution of a maximum clique with 49 vertices (5, 11, 14, 18, 20, 41, 43, 61, 65, 66, 69, 72, 87, 91, 93, 98, 131, 147, 157, 160, 167, 168, 177, 181, 188, 194, 199, 208, 210, 216, 235, 237, 240, 245, 253, 258, 260, 266, 269, 271, 284, 318, 323, 334, 339, 372, 381, 385, 386) and, in addition, displays information, scanned from the GAMS environment on the best solution having a value of 55. Therefore, the current solution differs from the optimum by 6. See Figure 1.

```
MIP Solution:           49.000000    (877406 iterations, 11107 nodes)
Final Solve:            49.000000    (0 iterations)

Best possible:          55.000000
Absolute gap:            6.000000
Relative gap:            0.109091
```

Figure 1. Scanned result from GAMS.

The remaining part of the paper is organized as follows. Section 2 presents the basic concepts and the first of the theoretical models, which we, however, do not use due to the complex two-phase calculation. Instead, in Section 3 we focus on two single-phase computation models, their modifications for direct use in GAMS and evaluations of their theoretical time complexities, including implementation details along with code writing. Section 4 presents computational results on a set of DIMACS benchmarks. Section 5 is a final summary with an indication of the direction for further research.

2. Basic Notions

Since the basic algorithm for finding the maximum clique in a graph is related to the notion of an independent set, we start with it.

A set of vertices in a graph G is *independent* if no two of its vertices are connected by an edge. An independent set S of vertices in a graph G is called a *maximal independent set* if S is not a proper subset of any other independent set of vertices of G. This means that no vertex can be added to a maximal independent set without it ceasing to be independent.

The *largest independent set* is the maximal independent set that has the largest number of vertices. The number of vertices in the largest independent set is called the *independence number* of G and is denoted by $\alpha(G)$.

A *clique* in a graph G is a subgraph of G that is a complete graph. A *maximal clique* is a clique that cannot be extended by including one more adjacent vertex. A *maximum clique of*

a graph G is a clique, such that there is no clique with more vertices. The *clique number* of a graph G, denoted by $\omega(G)$, is the number of vertices in a maximum clique of G.

If we construct a so-called *complement graph* G' of G in which two vertices are connected by an edge exactly when they are not connected in the original graph G, then it is clear that the cliques in the graph G correspond to the maximal independent sets in the complement graph G'.

Let $G = (V, E)$ be a finite simple graph, where $V = \{v_1, \ldots, v_n\}$, C be a set of vertices in a clique, I_s^* denote the set of all maximal independent sets in G, and $x_1, \ldots, x_n$ denote decision variables, where $x_i = 1$ if $v_i \in C$ and $x_i = 0$ if $v_i \notin C$.

Then, the maximum clique problem can be formulated as the following integer programme [65]:

$$z = \sum_{i=1}^{n} x_i \rightarrow \max \tag{1}$$

subject to

$$\sum_{i \in I_s} x_i \leq 1, \ \forall I_s \in I_s^* \tag{2}$$

$$x_i \in \{0, 1\}, \ i = 1, \ldots, n \tag{3}$$

Other algorithms using independent sets for specific constraints are described in [77,78].

A specific case of the Maximum Clique Problem is the Maximum Weight Clique Problem [68,79,80], where vertices v_i are assigned weights w_i and the objective is to find the clique that has the maximum sum of the weights. The objective function (1) changes to (4):

$$z = \sum_{i=1}^{n} w_i x_i \rightarrow \max \tag{4}$$

The model determined by Equations (1)–(3) is simple, but its disadvantage is that the set of maximal independent sets must be determined first, so the calculation has two phases and is not suitable for direct calculation in GAMS.

3. Direct Models and Implementation Views in GAMS

In the sense of the last comment in the previous section, rather than the *independent set reformulation* of the maximum clique problem, expressed by Equations (1)–(3), the so-called *edge reformulation* is better for us.

3.1. The First Model

The first model is as follows [3,64]:

$$z = \sum_{i=1}^{n} x_i \rightarrow \max \tag{5}$$

subject to

$$x_i + x_j \leq 1, \ \forall \{v_i, v_j\} \notin E \tag{6}$$

$$x_i \in \{0, 1\}, \ i = 1, \ldots, n \tag{7}$$

We will adapt this model to be easily usable in the GAMS environment. We modify Equation (6) by multiplying $x_i + x_j$ by the complementary values of the matrix E where the values 0 and 1 are interchanged and exclude the elements of the main diagonal, i.e., $i \neq j$. This gives us the following form:

$$(x_i + x_j) E_{comp}(i, j) \leq 1, \ i \neq j, \ i = 1, \ldots, n, \ j = 1, \ldots, n \tag{8}$$

We can provide the same function even more simply without the complementary values of the matrix E using $|e_{ij} - 1|$ instead of $E_{comp}(i, j)$ because $|e_{ij} - 1| = 0$ for $e_{ij} = 1$, and $|e_{ij} - 1| = 1$ for $e_{ij} = 0$.

Thus, the modified model (5)–(7) will have the following form:

$$z = \sum_{i=1}^{n} x_i \rightarrow \max \tag{9}$$

subject to

$$(x_i + x_j)|e_{ij} - 1| \leq 1,\ i \neq j,\ i = 1, \ldots, n,\ j = 1, \ldots, n \tag{10}$$

$$x_i \in \{0, 1\},\ i = 1, \ldots, n \tag{11}$$

The use of the absolute value function in Equation (10) is incorrect from the point of view of linear and integer models because only addition and multiplication operations are allowed in constraints, but the absolute value function is available in the GAMS environment.

Now, we present the complete code of the maximum clique calculation in GAMS for the graph in Figure 2. The resulting solution is shown in Figure 3. The graph contains a large number of cliques with three vertices, one clique with four vertices (2, 3, 5, 13) and the maximum clique consists of vertices 5, 7, 8, 10 and 11.

```
$TITLE Maximum clique
$OFFSYMXREF
$OFFUELLIST
$OFFUELXREF

* section defining indexes
SETS
  I vertices  /1*21/;
  ALIAS(J,I);

* input data section

TABLE E(I,J)
       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
  1    0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  2    1  0  1  0  1  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0
  3    1  1  0  1  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
  4    1  0  1  0  1  1  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0
  5    0  1  1  1  0  1  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0
  6    0  0  0  1  1  0  1  1  0  1  1  1  1  0  0  0  0  0  0  0  0
  7    0  0  0  0  1  1  0  1  0  1  1  1  0  0  0  0  0  0  0  0  0
  8    0  0  0  1  0  1  1  0  1  1  1  0  0  0  0  0  0  0  0  0  0
  9    0  0  0  1  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0
 10    0  0  0  0  0  1  1  1  1  0  1  0  0  0  0  0  0  0  1  0  0
 11    0  0  0  0  0  1  1  1  0  1  0  1  0  0  1  0  0  1  0  0  1
 12    0  0  0  0  1  0  1  0  0  0  1  0  1  0  1  1  0  0  0  0  0
 13    0  1  1  0  1  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0
 14    0  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0
 15    0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  1  1  1  1  0  0
 16    0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  0  1  0  0  0  0
 17    0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  1  0  0
 18    0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  1  1  1
 19    0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  1  0  1  0
 20    0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  1  0  1
 21    0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  1  0  1  0;
```

```
* variables section (decision variables and objective function)
VARIABLES
  X(I)
  Cmax objective function (number of vertices in the maximum clique);
  BINARY VARIABLE X;

*section describing the system of (in)equalities
EQUATIONS
  EQ1(I,J)
  OBJFCE number of vertices in the maximum clique;

  EQ1(I,J)$(ORD(I) NE ORD(J)) .. (X(I)+X(J))*ABS(E(I,J)-1) =L= 1;
  OBJFCE .. Cmax =E= SUM(I,X(I));

*description of the model, running the solver, and displaying the results
MODEL CLIQUE /ALL/;
SOLVE CLIQUE USING MIP MAXIMIZING Cmax;
* DISPLAY NN, X.L, Cmax.L;
DISPLAY X.L, Cmax.L;
```

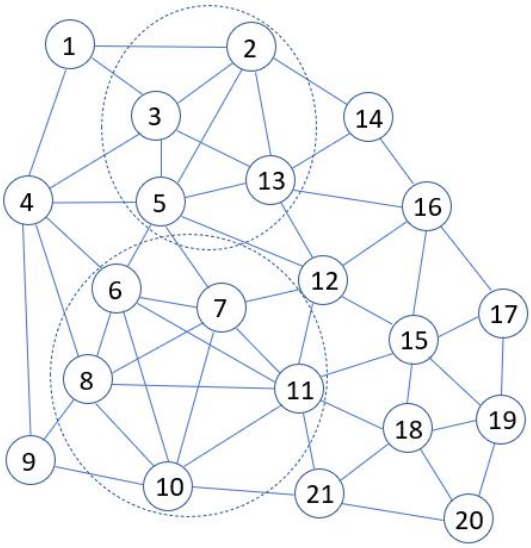

Figure 2. Graph with 21 vertices.

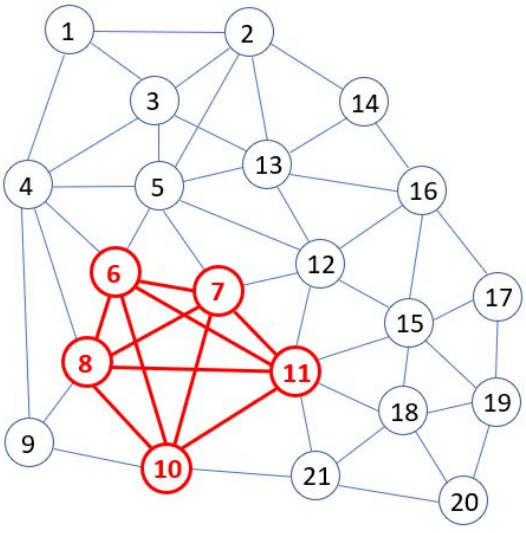

Figure 3. Maximum clique for the graph from Figure 2.

3.1.1. Time Complexity of the First Model

Theorem 1. *The modified model (9)–(11) runs in $\mathcal{O}(4^n)$ time.*

Proof. The size of the search space is determined by the number of all assignments of 0 and 1 to x_i and x_j in Equation (10). Since the combination numbers $\binom{n}{k}$, $k = 0, 1, \ldots, n$ express how many ways k zeros can be placed in a vector of n elements, according to the binomial theorem, the number of all assignments of zeros to x_i is given by

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n-1} + \binom{n}{n} = (1+1)^n = 2^n \tag{12}$$

The same situation is for x_j. However, we must exclude from these combinations the cases where $x_i = x_j$, i.e., both variables are either 0 or 1.

Hence, we get the size of the search space:

$$2^n 2^n - n - n = 2^{2n} - 2n = 4^n - 2n = \mathcal{O}(4^n) \tag{13}$$

□

3.2. The Second Model

For a vertex v_j of G, we define the set $N_N(j)$ which contains all non-neighbours of $v_j \in G$. Although vertex v_j is not adjacent to itself, v_j is not considered to be an element of $N_N(j)$. The cardinality of $N_N(j)$ is denoted by h_j. More precisely, $h_j = |N_N(j)|$ if $N_N(j) \neq \emptyset$ and $h_j = 1$ if $N_N(j) = \emptyset$. That means $h_j = \max\{1, |N_N(j)|\}$.

In [66], the following model was derived.

$$z = \sum_{i=1}^{n} x_i \to \max \tag{14}$$

subject to

$$h_j x_j + \sum_{i \in N_N(j)} x_i \leq h_j, \; j = 1, \ldots, n \tag{15}$$

$$x_i \in \{0, 1\}, \; i = 1, \ldots, n \tag{16}$$

First, instead of sets $N_N(j)$ which contain all non-neighbours of $v_j \in G$, we introduce the set $N_N(i, j)$, where $N_N(i, j) = 1$ if $E(i, j) = 0$ and $N_N(i, j) = 0$ if $E(i, j) = 1$.

The elements $N_N(i, i)$, $i = 1, \ldots, n$ are assigned the values 0 and the values of h_i must be reduced by 1 because the values of $N_N(i, i)$, $i = 1, \ldots, n$ were set to 1 during the first initialization.

The final determination of h_i values by the equation $h_i = \max\{1, |N_N(i, j)|\}$ is similar to the previous case.

In GAMS code, we express this with a program section as follows:

```
LOOP(I,
       H(I)=0;
       LOOP(J,
              IF (E(I,J)=0,
                    NN(I,J)=1;
                    H(I)=H(I)+1;
                 ELSE
                    NN(I,J)=0;
                );
           );
    );

LOOP(I,
       NN(I,I)=0;
```

```
        H(I)=H(I)-1;
        IF (H(I)=0,
             H(I)=1;
           );
      );
```

In Equation (15), instead of traversing the original sets $N_N(j)$, we multiply x_i by the newly introduced elements of the set $(N_N(i,j)$ and get:

$$h_j x_j + \sum_{i=1}^{n} (x_i N_N(i,j)) \leq h_j,\ j = 1, \ldots, n \tag{17}$$

It is clear that, after redefining $N_N(j)$ to $N_N(i,j)$, the matrix N_N is complementary to E, i.e., the values 1 and 0 are interchanged so there is no need to introduce the matrix N_N and we can directly overwrite the elements of the matrix E with complementary values. However, the elements of the main diagonal remain zero. In the modified model (20)–(22), $N_N(i,j)$ is replaced by $E_{comp}(i,j)$, where the matrix E_{comp} represents the rewritten values of the matrix E.

In the GAMS code, we create new values of the matrix E in the upper triangular matrix and then copy them to the lower triangular matrix since the matrix E is symmetric. Thus, Equation (17) changes to (18).

$$h_j x_j + \sum_{i=1}^{n} (x_i E_{comp}(i,j)) \leq h_j,\ j = 1, \ldots, n \tag{18}$$

Since the initial part of the code in GAMS is identical to the above code, we only show the part from the PARAMETERS section, where the matrix E is overwritten and the values of h_j are calculated, and then the EQUATIONS section is also modified.

```
PARAMETERS
H(J) cardinality of non-neighbours set of vertex j;

*inverse in upper triangular matrix and copy to lower triangular matrix
LOOP(I,
       H(I)=0;
       LOOP(J$(ORD(J)>ORD(I)),
              IF (E(I,J)=0,
                    E(I,J)=1;
                    E(J,I)=1;
                    H(I)=H(I)+1;
                 ELSE
                    E(I,J)=0;
                    E(J,I)=0;
                 );
           );
    );

LOOP(I,
       LOOP(J$(ORD(J)<ORD(I)),
              IF (E(I,J)=1,
                 H(I)=H(I)+1;
                 );
           );
    );

LOOP(I,
       E(I,I)=0;
       IF (H(I)=0,
```

```
            H(I)=1;
          );
     );

* variables section (decision variables and objective function)
VARIABLES
  X(I)
  Cmax objective function (number of vertices in the maximum clique);
  BINARY VARIABLE X;

*section describing the system of (in)equalities
EQUATIONS
  EQ1(J)
  OBJFCE number of vertices in the maximum clique;

  EQ1(J) .. H(J)*X(J)+SUM(I,X(I)*E(I,J)) =L= H(J);
  OBJFCE .. Cmax =E= SUM(I,X(I));

*description of the model, running the solver, and displaying the results
MODEL CLIQUE /ALL/;
SOLVE CLIQUE USING MIP MAXIMIZING Cmax;
DISPLAY X.L, Cmax.L;
```

Overwriting the matrix E in the programme, we still need the original values to draw the result, but the original data for larger graphs are usually incorporated into the code from external files (benchmarks) and are still available there. In addition, GAMS has no graphical tools and the results of the calculations are only in the text form. This requires exporting them to a suitable programme and postprocessing.

Using the absolute value function, Equation (18) can be modified without rewriting the matrix E similarly to Equation (10) as follows:

$$h_j x_j + \sum_{i=1, i\neq j}^{n} x_i |e_{ij} - 1| \leq h_j,\ j = 1, \ldots, n \tag{19}$$

The complete model is then the following:

$$z = \sum_{i=1}^{n} x_i \to \max \tag{20}$$

$$h_j x_j + \sum_{i=1, i\neq j}^{n} x_i |e_{ij} - 1| \leq h_j,\ j = 1, \ldots, n \tag{21}$$

$$x_i \in \{0, 1\},\ i = 1, \ldots, n \tag{22}$$

The corresponding code in GAMS from the PARAMETERS section will be as follows:

```
PARAMETERS
H(J) cardinality of non-neighbours set of vertex j;

LOOP(I,
       H(I)=0;
       LOOP(J$(ORD(I) NE ORD(J)),
                   H(I)=H(I)+1;
           );
    );

LOOP(I,
       IF (H(I)=0,
```

```
            H(I)=1;
          );
     );

* variables section (decision variables and objective function)
VARIABLES
  X(I)
  Cmax objective function (number of vertices in the maximum clique);
  BINARY VARIABLE X;

*section describing the system of (in)equalities
EQUATIONS
  EQ1(J)
  OBJFCE number of vertices in the maximum clique;
  EQ1(J) .. H(J)*X(J)+SUM(I$(ORD(I) NE ORD(J)),X(I)*ABS(E(I,J)-1)) =L= H(J);
  OBJFCE .. Cmax =E= SUM(I,X(I));

*description of the model, running the solver, and displaying the results
MODEL CLIQUE /ALL/;
SOLVE CLIQUE USING MIP MAXIMIZING Cmax;
* DISPLAY NN, X.L, Cmax.L;
DISPLAY X.L, Cmax.L;
```

Time Complexity of the Second Model

Theorem 2. *The modified model (20)–(22) runs in* $\mathcal{O}(n4^n)$ *time.*

Proof. Assuming that we already know the values of h_j, the time complexity of model (20)–(22) follows from Equation (21) and reasoning in much the same way as in Section 3.1.1; it is given by Equation (23):

$$2^n((n-1)2^n) = (n-1)4^n = \mathcal{O}(n4^n), \tag{23}$$

where $n-1$ in $2^n((n-1)2^n)$ expresses the summation operation for $n-1$ elements in Equation (23). Since the computation of h_j with increasing n grows more slowly than $(n-1)4^n$, the total time complexity of the second algorithm is given by $\mathcal{O}(n4^n)$. □

It is clear that the time complexity of the second model is greater than that of the first one.

4. MCP Computational Results

Before we move on to the GAMS calculations for both models, we show the theoretical time requirements of these models for a 2.1 GHz processor. For simplicity, we use approximate complexities from the $\mathcal{O}$ notation. These times are only approximate because the evaluation of Equations (10) and (21) is not performed during a single machine operation. However, they give an idea of whether problems of a certain size can be solved in an acceptable time.

In Table 1, for several values of n, the theoretical computation times are given.

From Table 1 it can be seen that, by brute force, the maximum clique of a graph can be calculated in a reasonable amount of time for 20 vertices, but already for 30 vertices, the computation time is extremely long.

As for the time complexity of the computation in GAMS, it cannot be formally expressed exactly in $\mathcal{O}$ notation because the implementation of the GAMS computational kernel is not freely available.

The ability of computing optimal solutions in GAMS was checked using standard benchmarks [81].

In the GAMS programme, unlike the sample code in Section 3, the benchmark data are not loaded directly into the `TABLE E(I,J)` structure, but, due to the size, they are included from an external file by the `$INCLUDE externalfilename` statement.

Table 1. Theoretical $\mathcal{O}$ time complexities for a 2.1 GHz processor.

n	4^n	Time	$n4^n$	Time
10	1,048,576	0.00049 s	10,485,760	0.0049 s
20	1.09951×10^{12}	8 min 43.5 s	2.19902×10^{13}	2 h 54 min 31 s
30	1.15292×10^{18}	174 years 32 days 20 h	3.45876×10^{19}	5224 years 255 days 1 h
40	1.20893×10^{24}	$1.825466013 \times 10^{8}$ years	4.83570×10^{25}	$7.301864050 \times 10^{9}$ years
50	1.26765×10^{30}	1.91414×10^{14} years	6.33825×10^{31}	9.5707×10^{15} years
60	1.32923×10^{36}	2.00712×10^{20} years	7.97537×10^{37}	1.204727×10^{22} years
70	1.39380×10^{42}	2.10462×10^{26} years	9.75658×10^{43}	1.47323×10^{28} years

First, the matrix E needs to be reset in a double loop, because only those vertex pairs that are connected by an edge are listed in the benchmarks.

However, the benchmark data must be converted to GAMS syntax. For example, the data row `e 1 36`, since the matrix E is symmetric, needs to be converted to two assignment statements:

```
E("1","36")=1;
E("36","1")=1;
```

The comment lines in the benchmarks that do not correspond to the expression of vertex pairs connected by an edge are converted into the modified file as comments, i.e., the * symbol is inserted at the beginning of the corresponding lines and then they are ignored by GAMS during compilation.

The corresponding part of the code starting with the declaration `TABLE E(I,J)` and the specific location and name of the converted benchmark will be as follows:

```
TABLE E(I,J);
LOOP(I,
       LOOP(J,
             E(I,J)=0;
           );
    );
$INCLUDE C:\Txt\ARTICLES\2023_Symmetry_(MDPI)\benchmarks_clique_for_GAMS\anna.txt
```

The computational results for both models are summarised in Table 2. It can be seen that GAMS can get the best known solution for graphs up to 300 vertices and tens of thousands of edges in short times on a laptop with the parameters mentioned above.

Acceptable results were also obtained for much larger instances and for some of them even the best known values, e.g., for p_hat700-1, p_hat700-2 and p_hat700-3 instances with 700 vertices, for keller5 instance with 776 vertices and for hamming10-4 instance with 1024 vertices, almost all of them with hundreds of thousands of edges.

Where the best known solution has not been achieved, according to the theoretical assumptions, it is confirmed that the first model is more efficient and mostly finds a better solution in less time. However, this is not always satisfied; sometimes, the second model gives better results within the same calculation time limit when the final solution has not been reached. The intermediate results obtained depend on the way the GAMS program traverses the search space, the models used and the input data. Therefore, the relationship between the computation times of the two models may be different from the theoretical upper bounds on the time complexity of $\mathcal{O}$. It makes sense to use both models. This is a similar situation to the No Free Lunch Theorem [82,83] for heuristics, saying that any of the heuristics gives better results than other ones for all problems and all instances.

In addition, some instances may have multiple solutions with the same maximum clique number and therefore it is suitable to use more than one model.

Non-optimal results can be improved by extending the computation time limit, e.g., to 3000 s by the statement `OPTION ResLim = 3000`, as can also be seen in Table 2.

Table 2. Computational results for time limit 1000 s, * = time limit 3000 s, ** = time limit 5000 s, A1 = model (9)–(11), A2 = model (20)–(22).

Benchmark	$\|V\|$	$\|E\|$	A1 $\omega(G)$	A1 Time [s]	A2 $\omega(G)$	A2 Time [s]	Best Known
C125.9	125	6963	34	1.19	34	2.72	34
keller4	171	9435	11	12.33	11	6.64	11
brock200_2	200	9876	12	37.69	12	53.58	12
brock200_4	200	13,089	17	243.13	17	76.95	17
gen200_p0.9_44	200	17,910	44	0.28	44	0.92	44
gen200_p0.9_55	200	17,910	55	0.31	55	0.42	55
C250.9	250	27,984	44	906.55	44	26.33	44
hamming8-4	256	20,864	16	0.75	16	3.44	16
p_hat300-1	300	10,933	8	233.16	8	79.92	8
p_hat300-2	300	21,928	25	8.69	25	905.09	25
p_hat300-3	300	33,390	36	30.05	34	1.33	36
MANN_a27	378	70,551	123	0.20	124	0.55	126
brock400_2	400	59,786	23	24.98	23	926.86	29
brock400_4	400	59,786	22/23 **	54.94/5894.25	23	108.06	33
gen400_p0.9_55	400	71,820	49/50 *	55.06/2727.02	50	310.03	55
gen400_p0.9_65	400	71,820	49	132.30	64	380.30	65
gen400_p0.9_75	400	71,820	75	5.05	73	43.75	75
DSJC500_5	500	125,248	12	13.19	12	283.13	13
p_hat700-1	700	60,999	9	43.56	9	303.47	11
p_hat700-2	700	121,728	44	751.73	44	597.69	44
p_hat700-3	700	183,010	62	72.08	62	977.22	62
keller5.clq	776	225,990	27	83.83	27	996.11	27
brock800_2	800	208,166	19	87.22	18	217.27	24
brock800_4	800	208,166	18	583.94	17	62.88	26
C1000.9	1000	450,079	57	96.09	59	199.25	68
DSJC1000_5	1000	499,652	12/13 *	158.06/1332.08	12	130.56	15
hamming10-4	1024	434,176	40	499.59	40	934.28	40
MANN_a45	1035	533,115	327	0.41	330	0.34	345
p_hat1500-1	1500	284,923	11	436.95	10	520.81	12
p_hat1500-2	1500	568,960	64	204.11	56/64 **	336.88/4779.20	65
p_hat1500-3	1500	847,244	91	924.05	87	446.03	94
C2000.5	2000	999,836	0/13 *	0.00/2607.22	11/14 *	470.36/1434.59	16
C2000.9	2000	1,799,532	53/63 *	610.42/1028.58	63	538.25	80

5. Conclusions

This paper explores the maximum clique problem. The introductory section provides a detailed review of literature sources relevant to the problem. Among the possible computational approaches using approximate methods, stochastic heuristics, exact methods, and solutions based on mathematical models and their implementation in a suitable programming environment, it focuses on the last one.

The selected models were modified to be directly usable in GAMS; time complexities were derived. Their implementation also took into account the aspect of minimizing data structures, i.e., minimising the spatial complexity. The use of absolute values in the input matrix rewrite eliminated the need to compute a complementary adjacency matrix. The source codes presented may be an aid because this tool is not yet as well known as the MATLAB Optimisation Toolbox. The results obtained on a representative set of benchmarks document the applicability to instances with hundreds of vertices and tens of thousands of edges, where optimal solutions are obtained within a reasonable amount of time. The applicability range of GAMS for the maximum clique problem is much larger than that of the related graph colouring problem described in [84].

Even for extremely large instances with a thousand vertices and hundreds of thousands of edges, GAMS can be used. After exceeding a chosen time limit, the current solution can be used as an approximation of the optimum, similar to stochastic heuristics with a fixed number of iterations.

In future research, we expect to focus on the Maximum Weight Clique Problem.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in [81].

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

VLSI	Very-Large-Scale Integration
MCP	Maximum Clique Problem
GAMS	General Algebraic Modelling System

References

1. Chartrand, G.; Oellermann, O.R. *Applied and Algorithmic Graph Theory*; McGraw Hill: New York, NY, USA, 1993.
2. Diestel, R. *Graph Theory*; Springer: Berlin/Heidelberg, Germany, 2005.
3. Gutin, G. Independent Sets and Cliques. In *Handbook of Graph Theory*; Gross, J.L., Yellen, J., Eds.; CRC Press: Boca Raton, FL, USA, 2004; pp. 389–402.
4. Michaels, J.G.; Rosen, K.H. *Applications of Discrete Mathematics*; McGraw Hill: New York, NY, USA, 1991.
5. van Steen, M. Graph Theory and Complex Networks: An Introduction. 2010. Available online: https://pages.di.unipi.it/ricci/book-watermarked.pdf (accessed on 1 September 2023).
6. Cong, J.; Smith, L.M. A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VSLI Design. In Proceedings of the 30th ACM/IEEE International Design Automation Conference, Dallas, TX, USA, 14–18 June 1993; pp. 755–760.
7. Vorobev, S.; Kolosnitsyn, A.; Minarchenko, I. Determination of the Most Interconnected Sections of Main Gas Pipelines Using the Maximum Clique Method. *Energies* **2022**, *15*, 501. [CrossRef]
8. Samudrala, R.; Moult, J. Graph-theoretic Algorithm for Comparative Modeling of Protein Structure. *J. Mol. Biol.* **1998**, *279*, 287–302. [CrossRef] [PubMed]
9. Spirin, V.; Mirny, L.A. Protein Complexes and Functional Modules in Molecular Networks. *Biophys. Comput. Biol.* **2003**, *100*, 12123–12128. [CrossRef] [PubMed]
10. Liu, S.; Song, Y.; Zhang, F.; Feng, D.; Fulham, M.; Cai, W. Clique Identification and Propagation for Multimodal Brain Tumor Image Segmentation. In Proceedings of the International Conference Brain Informatics and Health, Omaha, NE, USA, 13–16 October 2016; pp. 755–760.
11. Reimann, M.W.; Nolte, M.; Scolamiero, M.; Turner, K.; Perin, R.; Chindemi, G.; Dłotko, P.; Levi, R.; Hess, K.; Markram, H. Cliques of Neurons Bound into Cavities Provide a Missing Link between Structure and Function. *Front. Comput. Neurosci.* **2017**, *11*, 48. [CrossRef] [PubMed]
12. Kato, Z. Markov Random Fields in Image Segmentation. In Proceedings of the 8th Conference on Signal, Speech and Image Processing, Vienna, Austria, 13–15 February 2008.
13. Ouyang, G.; Dey, D.P.; Zhang, P. Clique-Based Method for Social Network. *J. Classif.* **2020**, *37*, 254–274. [CrossRef]
14. Cavique, L.; Mendes, A.; Santos, J. Clique Communities in Social Networks. In *Quantitative Modelling in Marketing and Management*; Moutinho, L., Huarng, K.-H., Eds.; World Scientific Publishing: Singapore, 2012; pp. 469–490.
15. Boubaker, H.; Karoui, W. Improved Overlapping Community Detection in Networks based on Maximal Cliques Enumeration. *Procedia Comput. Sci.* **2020**, *176*, 858–867. [CrossRef]
16. Ballard, D.H.; Brown, C.M. *Computer Vision*; Prentice Hall: Englewood, NJ, USA, 1982.
17. Srinivasan, A. On the Approximability of Clique and Related Maximization Problems. *J. Comput. Syst. Sci.* **2003**, *67*, 633–651. [CrossRef]
18. Sukegawa, N.; Miyauchi, A. A Note on the Complexity of the Maximum Edge Clique Partitioning Problem with Respect to the Clique Number. *Discret. Optim.* **2013**, *10*, 331–332. [CrossRef]
19. Brause, C.; Randerath, B.; Rautenbach, D.; Schiermeyer, I. A Lower Bound on the Independence Number of a Graph in Terms of Degrees and Local Clique Sizes. *Discret. Appl. Math.* **2016**, *370*, 229–239. [CrossRef]
20. Budinich, M. Exact Bounds on the Order of the Maximum Clique of a Graph. *Discret. Appl. Math.* **2003**, *127*, 535–543. [CrossRef]
21. Alon, N.; Lingas, A.; Wahlen, M. Approximating the Maximum Clique Minor and Some Subgraph Homeomorphism Problems. *Theor. Comput. Sci.* **2007**, *374*, 149–158. [CrossRef]

22. Punnen, A.P.; Zhang, R. Analysis of an Approximate Greedy Algorithm for the Maximum Edge Clique Partitioning Problem. *Discret. Optim.* **2012**, *9*, 205–208. [CrossRef]
23. Gendron, B.; Hertz, A.; St-Louis, P. A Sequential Elimination Algorithm for Computing Bounds on the Clique Number of a Graph. *Discret. Optim.* **2008**, *5*, 615–628. [CrossRef]
24. McIntyre, R.; Soltys, M. An Improved Upper Bound and Algorithm for Clique Covers. *J. Discret. Algorithms* **2018**, *48*, 42–56. [CrossRef]
25. Gupta, S.K.; Singh, D.P. CBLA: A Clique Based Louvain Algorithm for Detecting Overlapping Community. *Procedia Comput. Sci.* **2023**, *218*, 2001–2009. [CrossRef]
26. Foucaud, F.; Mishra, S.; Narayanan, N.; Naserasr, R.; Valicov, P. Cliques in Exact Distance Powers of Graphs of Given Maximum Degree. *Procedia Comput. Sci.* **2021**, *195*, 427–436. [CrossRef]
27. Alcón, L.; Faria, L.; de Figueiredo, C.M.H.; Gutierrez, M. Split Clique Graph Complexity. *Theor. Comput. Sci.* **2013**, *506*, 29–42. [CrossRef]
28. Pattillo, J.; Veremyev, A.; Butenko, S.; Boginski, V. On the Maximum Quasi-Clique Problem. *Discret. Appl. Math.* **2013**, *161*, 244–257. [CrossRef]
29. Frankl, P.; Liu, E.L.L.; Wang, J. On the Maximum Number of Edges in Hypergraphs with Fixed Matching and Clique Number. *Eur. J. Comb.* **2022**, *106*. 103589. [CrossRef]
30. Garstka, M.; Cannon, M.; Goulart, P. A Clique Graph Based Merging Strategy for Decomposable SDPs. *IFAC-PapersOnLine* **2020**, *53*, 7355–7361. [CrossRef]
31. Brandstädt, A.; Hoang, C. On Clique Separators, nearly Chordal Graphs, and the Maximum Weight Stable Set Problem. *Theor. Comput. Sci.* **2007**, *389*, 295–306. [CrossRef]
32. Brandstädt, A.; Le, V.B.; Mahfud, S. New Applications of Clique Separator Decomposition for the Maximum Weight Stable Set Problem. *Theor. Comput. Sci.* **2007**, *370*, 229–239. [CrossRef]
33. Breugem, T.; Schlechte, T.; Schulz, C.; Borndörfer, R. A Three-Phase Heuristic for the Fairness-Oriented Crew Rostering Problem. *Comput. Oper. Res.* **2023**, *154*. 106186. [CrossRef]
34. Wu, B.; Peng, Y. The Connection between Polynomial Optimization, Maximum Cliques and Turán Densities. *Discret. Appl. Math.* **2017**, *225*, 114–121. [CrossRef]
35. Kashiwabara, K.; Okamoto, Y.; Uno, T. Matroid Representation of Clique Complexes. *Discret. Appl. Math.* **2007**, *155*, 1910–1929. [CrossRef]
36. Szabó, S.; Zavalnij, B. Decomposing Clique Search Problems into Smaller Instances Based on Node and Edge Colorings. *Discret. Appl. Math.* **2018**, *242*, 118–129. [CrossRef]
37. Lee, C.M. Algorithmic Aspects of Some Variations of Clique Transversal and Clique Independent Sets on Graphs. *Algorithms* **2021**, *14*, 22. [CrossRef]
38. Lee, C.M. Remarks on Parameterized Complexity of Variations of the Maximum-Clique Transversal Problem on Graphs. *Symmetry* **2022**, *14*, 676. [CrossRef]
39. Ghosh, S.K.; Shermer, T.C.; Bhattacharya, B.K.; Goswami, P.P. Computing the Maximum Clique in the Visibility Graph of a Simple Polygon. *J. Discret. Algorithms* **2007**, *5*, 524–532. [CrossRef]
40. Gollin, J.P.; Hendrey, K.; Methuku, A.; Tompinks, C.; Zhang, X. Counting Cliques in 1-Planar Graphs. *Eur. J. Comb.* **2023**, *109*, 103654. [CrossRef]
41. Hedman, B. The Maximum Number of Cliques in Dense Graphs. *Discret. Math.* **1985**, *54*, 161–166. [CrossRef]
42. Luo, R. The Maximum Number of Cliques in Graphs without Long Cycles. *J. Comb. Theory Ser. B* **2018**, *128*, 219–226. [CrossRef]
43. Bourgeois, N.; Croce, F.D.; Escoffier, B.; Paschos, V.T. Algorithms for Dominating Clique Problems. *Theor. Comput. Sci.* **2012**, *459*, 77–88. [CrossRef]
44. Kratsch, D.; Liedloff, M. An Exact Algorithm for the Minimum Dominating Clique Problem. *Theor. Comput. Sci.* **2007**, *385*, 226–240. [CrossRef]
45. Hsu, W.L.; Nemhauser, G.L. Algorithms for Minimum Covering by Cliques and Maximum Clique in Claw-Free Perfect Graphs. *Discret. Math.* **1981**, *37*, 181–191. [CrossRef]
46. Alcón, L. Clique-Critical Graphs: Maximum Size and Recognition. *Discret. Appl. Math.* **2006**, *154*, 1799–1802. [CrossRef]
47. Szabó, S.; Zaválnij, B. Clique Search in Graphs of Special Class and Job Shop Scheduling. *Mathematics* **2022**, *10*, 697. [CrossRef]
48. Karp, R.M. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*; Miller, R.E., Thatcher, J.W., Eds.; Plenum: New York, NY, USA, 1972; pp. 85–103.
49. Agrawal, P.; Abutarboush, H.F.; Ganesh, T.; Mohamed, A.W. Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009–2019). *IEEE Access* **2021**, *9*, 26766–26791. [CrossRef]
50. Battiti, R.; Protasi, M. Reactive Local Search for the Maximum Clique Problem. *Algorithmica* **2001**, *29*, 610–637. [CrossRef]
51. Smith, D.H.; Montemanni, R.; Perkins, S. The Use of an Exact Algorithm within a Tabu Search Maximum Clique Algorithm. *Algorithms* **2020**, *13*, 253. [CrossRef]
52. Lewis, R.M.R. *A Guide to Graph Colouring. Algorithms and Applications*; Springer: Berlin/Heidelberg, Germany, 2016.
53. Hansen, P.; Mladenović, N.; Urošević, D. Variable Neighborhood Search for the Maximum Clique. *Discret. Appl. Math.* **2004**, *145*, 117–125. [CrossRef]

54. Manrique, D.; Rodríguez-Patón, A.; Sosík, P. On the Scalability of Biocomputing Algorithms: The Case of the Maximum Clique Problem. *Theor. Comput. Sci.* **2011**, *412*, 7075–7086. [CrossRef]
55. Singh, K.K.; Govinda, L. A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VSLI Design. In Proceedings of the 8th International Conference on Intelligent Systems and Control, Coimbatore, India, 10–11 January 2014; pp. 269–273.
56. Thampi, S.M.; Krishna, M.P. *A Fast Heuristic Algorithm Based on Verification and Elimination Methods for Maximum Clique Problem*; Report; College of Engineering: Kasaragod, India, 2007; 5p. Available online: https://arxiv.org/ftp/arxiv/papers/0710/0710.0748.pdf (accessed on 1 September 2023).
57. Yu, Z.; Wang, X.; Zhao, J. Local Dense Decision-Making Method for Solving the Maximum Clique Problem in Large-Scale Undirected Graphs. *Comput. Electr. Eng.* **2022**, *102*. 108220. [CrossRef]
58. Balash, V.; Stepanova, A.; Volkov, D.; Mironov, S.; Faizliev, A.; Sidorov, S. Testing a Heuristic Algorithm for Finding a Maximum Clique on DIMACS and Facebook Graphs. *WSEAS Trans. Syst. Control* **2010**, *15*, 93–101. [CrossRef]
59. Bomze, I.M.; Budinich, M.; Pardalos, P.M.; Pelillo, M. The Maximum Clique Problem. In *Handbook of Combinatorial Optimization*; Du, D.-Z., Pardalos, P.M., Eds.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1999; Volume A, pp. 1–74.
60. Scozzari, A.; Tardella, F. A Clique Algorithm for Standard Quadratic Programming. *Discret. Appl. Math.* **2008**, *156*, 2439–2448. [CrossRef]
61. Storch, T. Finding Large Cliques in Sparse Semi-Random Graphs by Simple Randomized Search Heuristics. *Theor. Comput. Sci.* **2007**, *386*, 114–131. [CrossRef]
62. Pelillo, M. Heuristics for Maximum Clique and Independent Set. In *Encyclopedia of Optimization*; Floudas, C., Pardalos, P.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1508–1520.
63. Bomze, I.M.; Budinich, M.; Pelillo, M.; Rossi, C. Annealed Replication: A New Heuristic for the Maximum Clique Problem. *Discret. Appl. Math.* **2002**, *121*, 27–49. [CrossRef]
64. Pardalos, P.M.; Xue, J. The Maximum Clique Problem. *J. Glob. Optim.* **1994**, *4*, 301–328. [CrossRef]
65. Kardos, D.; Patassy, P.; Szabó, S.; Zaválnij, B. Numerical Experiments with LP formulations of the Maximum Clique Problem. *Cent. Eur. J. Oper. Res.* **2022**, *30*, 1353–1367. [CrossRef]
66. Croce, F.D.; Tadei, R. A Multi-KP Modeling for the Maximum-Clique Problem. *Eur. J. Oper. Res.* **1994**, *73*, 555–561. [CrossRef]
67. Züge, A.P.; Carmo, R. On Comparing Algorithms for the Maximum Clique Problem. *Discret. Appl. Math.* **2018**, *247*, 1–13. [CrossRef]
68. Shimizu, S.; Yamaguchi, K.; Saitoh, T.; Masud, S. Fast Maximum Weight Clique Extraction Algorithm: Optimal Tables for Branch-and-Bound. *Discret. Appl. Math.* **2017**, *223*, 120–134. [CrossRef]
69. Li, C.M.; Jiang, H.; Manyà, F. On Minimization of the Number of Branches in Branch-and-Bound Algorithms for the Maximum Clique Problem. *Comput. Oper. Res.* **2017**, *84*, 1–15. [CrossRef]
70. Tomita, E.; Sutani, Y.; Higashi, T.; Takahashi, S.; Wakatsuki, M. A Simple and Faster Branch-and-Bound Algorithm for Finding a a Maximum Clique. *Lect. Notes Comput. Sci.* **2010**, *5942*, 191–203.
71. Östergard, P.R. A Fast Algorithm for the Maximum Clique Problem. *Discret. Appl. Math.* **2002**, *120*, 197–207. [CrossRef]
72. Segundo, P.S.; Furin, F.; Álvarez, D.; Pardalos, P.M. CliSAT: A New Exact Algorithm for Hard Maximum Clique Problems. *Eur. J. Oper. Res.* **2022**, *307*, 1008–1025. [CrossRef]
73. Prosser, P. Exact Algorithms for Maximum Clique: A Computational Study. *Discret. Optim.* **2012**, *5*, 545–587. [CrossRef]
74. Seda, P.; Seda, M.; Hosek, J. OnMathematical Modelling of Automated Coverage Optimization in Wireless 5G and beyond Deployments. *Appl. Sci.* **2020**, *10*, 8853. [CrossRef]
75. Seda, M. Steiner Tree Problem in Graphs and Mixed Integer Linear Programming-Based Approach in GAMS. *WSEAS Trans. Comput.* **2022**, *21*, 257–262. [CrossRef]
76. Seda, M. The Assignment Problem and Its Relation to Logistics Problems. *Algorithms* **2022**, *15*, 377. [CrossRef]
77. Durán, G.; Lin, M.C.; Mera, S.; Szwarcfiter, J.L. Algorithms for Clique-Independent Sets on Subclasses of Circular-Arc Graphs. *Discret. Appl. Math.* **2006**, *154*, 1783–1790. [CrossRef]
78. Čenek, E.; Stewart, L. Maximum Independent Set and Maximum Clique Algorithms for Overlap Graphs. *Discret. Appl. Math.* **2003**, *131*, 77–91. [CrossRef]
79. Busygin, S. A New Trust Region Technique for the Maximum Weight Clique Problem. *Discret. Appl. Math.* **2006**, *154*, 2080–2096. [CrossRef]
80. Gschwind, T.; Irnich, S.; Podlinski, I. Maximum Weight Relaxed Cliques and Russian Doll Search Revisited. *Discret. Appl. Math.* **2018**, *234*, 131–138. [CrossRef]
81. DIMACS: Clique Benchmark Instances. Report. 2020. Available online: http://iridia.ulb.ac.be/~fmascia/maximum_clique/ (accessed on 1 September 2023).
82. Wolpert, D.H.; McReady, W.G. No Free Lunch Theorems for Optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
83. Wolpert, D.H.; McReady, W.G. Coevolutionary Free Lunches. *IEEE Trans. Evol. Comput.* **2005**, *9*, 721–735. [CrossRef]
84. Seda, M. Integer Programming Approach to Graph Colouring Problem and Its Implementation in GAMS. *WSEAS Trans. Syst.* **2023**, *22*, 532–537. [CrossRef]