Optimization Letters
https://doi.org/10.1007/s11590-023-02076-8

ORIGINAL PAPER

A polytime preprocess algorithm for the maximum independent set problem

Samuel Kroger[1] · Hamidreza Validi[2] · Illya V. Hicks[1]

Received: 24 July 2023 / Accepted: 30 October 2023

Abstract
The maximum independent set (MIS) seeks to find a subset of vertices with the maximum size such that no pair of its vertices are adjacent. This paper develops a recursive fixing procedure that generalizes the existing polytime algorithm to solve the maximum independent set problem on chordal graphs, which admit simplicial orderings. We prove that the generalized fixing procedure is safe; i.e., it does not remove all optimal solutions of the MIS problem from the solution space. Our computational results show that the proposed recursive fixing algorithm, along with the basic mixed integer programming (MIP) of the MIS, outperforms the pure MIP formulation of the problem. Our codes, data, and results are available on GitHub.

Keywords Maximum independent set · Graph theory · Preprocessing

1 Introduction

Given a graph $G = (V, E)$ with vertex set V of size n and edge set E of size m, the maximum independent set (MIS) problem aims to find a largest subset of vertices in which no pair are adjacent. An independent set is also called a stable set, packing set, co-clique or anticlique. We note that any independent set in graph G corresponds to a clique in its complement (i.e., $\bar{G}$). Let $\alpha(G)$ and $\omega(\bar{G})$ denote the size of a maximum independent set in G and the size of the maximum clique in $\bar{G}$, respectively. Then, we have $\alpha(G) = \omega(\bar{G})$.

✉ Hamidreza Validi
hvalidi@ttu.edu

Samuel Kroger
sak8@rice.edu

Illya V. Hicks
ivhicks@rice.edu

[1] Computational Applied Mathematics and Operations Research, Rice University, Houston, USA

[2] Industrial, Manufacturing & Systems Engineering, Texas Tech University, Lubbock, USA

Published online: 24 November 2023

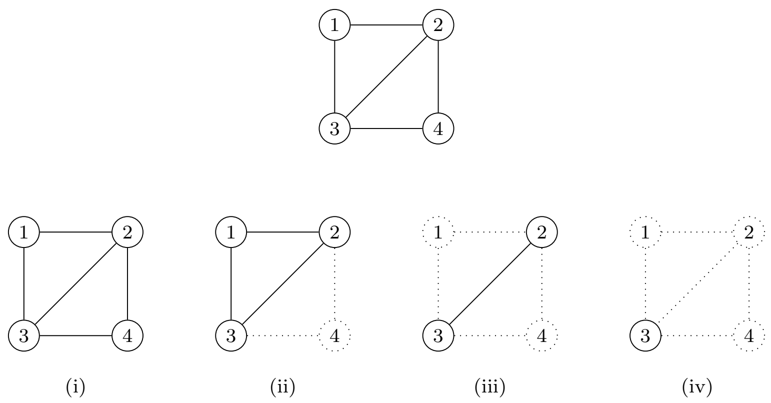

Fig. 1 A chordal graph (at the top) with a simplicial ordering of 4, 1, 2, 3 (at the bottom). At the bottom, (i) vertex 4 is a simplicial vertex in induced subgraph $G[\{4, 1, 2, 3\}]$, (ii) vertex 1 is a simplicial vertex in induced subgraph $G[\{1, 2, 3\}]$, (iii) vertex 2 is a simplicial vertex in induced subgraph $G[\{2, 3\}]$, and (iv) vertex 3 is a simplicial vertex in induced subgraph $G[\{3\}]$

The MIS problem is NP-hard [7] which means there is no polytime algorithm for solving the problem yet. There are classes of graphs for which the maximum independent set problem is solvable in polytime. Mannino et al. [12] propose a polytime algorithm for solving the maximum weighted independent set (MWIS) problem for a superclass of interval graphs. The MWIS problem can also be solved in polytime for claw-free graphs [5, 13, 14, 16, 19]. It is also polytime solvable on P_5-free graphs [11] and perfect graphs [8].

Frank [6] provides a polytime algorithm for solving the problem on chordal graphs. Furthermore, Bondy and Murty [1] provide a polytime algorithm (see Excercise 9.7.5) for solving the problem on chordal graphs based on a simplicial ordering. A vertex is simplicial if its neighbors form a clique in graph G. A simplicial ordering is an enumeration $v_1, v_2, \ldots, v_n$ of vertices if vertex v_i is a simplicial vertex of the induced graph $G[\{v_i, v_{i+1}, \ldots, v_n\}]$ for every $i \in \{1, \ldots, n\}$ [1]. The following corollary of Bondy and Murty [1] explains why the MIS is easy on chordal graphs.

Corollary 1 (Corollary 9.22 of Bondy and Murty [1]) *A graph is chordal if and only if it has a simplicial ordering.*

For example, Fig. 1 shows a chordal graph and its corresponding simplicial ordering. In this graph, we first add vertex 4 to the MIS and remove 4 and its neighbors (i.e., vertices 2 and 3) from the ordering. Then, we add vertex 1 (the only remaining vertex) to the MIS and we have set $\{1, 4\}$ as the MIS.

There are polytime algorithms for solving and preprocessing the MIS and the maximum clique problems on special graphs. Buchanan et al. [2] propose an algorithm for solving the maximum clique problem that is parameterized by the degeneracy of the graph. Their algorithm solves the maximum clique problem in $O(nm)$ for graphs with a degeneracy (i.e., a measure of graph sparsity) of at most $4\log_2^m + O(1)$. In their algorithm, they employ the algorithm of Robson [17] that

```
1: edges($\{a,b\}$) $\leftarrow$ False for every vertex pair $\{a,b\} \in \binom{V}{2}$
2: edges($\{a,b\}$) $\leftarrow$ True for every edge $\{a,b\} \in E$
3: simplicial($v$) $\leftarrow$ True for every vertex $v \in V$
4: for all $v \in V$ do
5:     for all $\{a,b\} \in \binom{N_G(v)}{2}$ do
6:         if edges($\{a,b\}$) = True then
7:             simplicial($v$) $\leftarrow$ False
8:             break
9: return simplicial
```

Algorithm 1: Simplicial (G)

solves the maximum independent set in $O(2^{n/4})$. Walteros and Buchanan [20] provide an algorithm that runs in $O(m^{1.5})$ for real-life graphs. They employ the relationship between the minimum vertex cover and the maximum clique problem to use multiple preprocessing algorithms, e.g., Buss kernelization [3], Nemhauser–Trotter kernelization [15], and crown reduction [10].

In this paper, we generalize the simplicial fixing idea on any general class of graphs. We apply the simplicial fixing idea until there is no more simplicial vertices in the graph. We finally solve the MIS on the remained graph (the core) via the following classical mixed integer programming (MIP) model in which for every vertex $v \in V$, binary decision variable x_v is one if vertex v belongs to an independent set.

$$\max \sum_{v \in V} x_v \tag{1a}$$

$$x_u + x_v \leq 1 \quad \forall e = \{u,v\} \in E \tag{1b}$$

$$x \in \{0,1\}^n. \tag{1c}$$

Our computational results show the superiority of the recursive fixing procedure over the pure MIP approach for a set of benchmark instances.

2 A recursive fixing procedure

This section presents a polytime procedure to fix the binary decision variable of the MIP formulation (1) to one or zero. For every vertex subset $Q \subseteq V$, let $N_G(Q)$ and $N_G[Q]$ be the open and closed neighborhood of Q, respectively. Formally, we define $N_G(Q) := \{w \in V \backslash Q : w \text{ is adjacent to some vertex } q \in Q\}$ and $N_G[Q] := Q \cup N_G(Q)$. For every vertex set $Q \subseteq V$, we also employ $G[Q]$ and $\binom{Q}{2}$ to show the subgraph induced by Q and all pairs of vertices in the set Q, respectively. At each iteration of the proposed procedure, we first find the simplicial vertices. Given graph $G = (V, E)$

```
1: I ← ∅
2: let cc($G[S]$) be the set of all connected components of $G[S]$
3: for all c ∈ cc($G[S]$) do
4:     let $v$ be a vertex of the component c
5:     I ← I ∪ {$v$}
6: return I
```

Algorithm 2: indepSimplicial (*G*, *S*)

at any iteration of our proposed procedure, Algorithm 1 finds the simplicial vertices of G in $O(nm)$ as we have at most n and m iterations in lines 4 and 5, respectively.

Let S be the set of simplicial vertices returned by Algorithm 1. Then, Algorithm 2 returns a maximum independent set of $G[S]$ in $O(m)$ as the complexity of finding connected components is $O(m)$.

The following proposition of Salemi and Buchanan [18] summarizes the complexity of finding a maximum independent set of simplicial vertices elaborated by Algorithms 1 and 2.

Proposition 1 (Proposition 4 of Salemi and Buchanan [18]) *Algorithms 1 and 2 find a maximum independent set of simplicial nodes in* $O(nm)$.

For every vertex $v \in D$, we set x_v to one and x_u to zero for every vertex u in the neighbor set of v. To our knowledge, no previous work considered simplicial fixing for the MIS problem on general graphs. Nemhauser and Trotter [15] propose an LP-based fixing rule for the problem. They prove that variables with binary values in an optimal continuous relaxation of formulation (1) get the same value in an optimal solution of the MIS formulation (1). Similarly, Hammer et al. [9] prove that if a binary variable of formulation (1) takes zero or one in all optimal solutions of the continuous relaxation of the formulation, then it takes the same value in all optimal binary solutions of formulation (1). Furthermore, Butenko and Trukhanov [4] employ the polytime solvable critical independent set problem as a preprocess for solving the MIS problem. Regarding the simplicial fixing approach, Salemi and Buchanan [18] propose a simplicial fixing for the distance-based critical node problem.

We define F_1 and F_0 as the set of vertices for which their corresponding decision variables in the MIP formulation (1) are set to one and zero, respectively. Algorithm 3 explains the recursive simplicial fixing idea.

In Algorithm 3, line 1 creates a temporary copy of the input graph. Line 2 initializes the set of vertices for which the decision variables x will be fixed to zero (F_0) and one (F_1). Line 3 defines the remove dictionary and sets every vertex not to be removed. Line 5 calls Algorithm 1 to compute the simplicial vertices of G'. Line 6 calls Algorithm 2 to find an independent set of the simplicial vertices in graph G'. Line 7 updates F_1 and F_0 using the independent set I obtained in line 6. Lines 8–10 update the temporary graph G' for the next iteration by removing

```
1: $G' = (V', E') \leftarrow G = (V, E)$
2: $F_0 \leftarrow \varnothing$ and $F_1 \leftarrow \varnothing$
3: remove($v$) $\leftarrow$ False for every vertex $v \in V$
4: do
5:     $S \leftarrow$ Simplicial($G'$) (see Algorithm 1)
6:     $I \leftarrow$ indepSimplicial($G', S$) (see Algorithm 2)
7:     $F_1 \leftarrow F_1 \cup I$ and $F_0 \leftarrow F_0 \cup N_{G'}(I)$
8:     remove($v$) $\leftarrow$ True for every vertex $v \in N_{G'}[I]$
9:     $R := \{v \in V(G') : \text{remove}(v) = \text{False}\}$
10:    $G' \leftarrow G'[R]$
11: while $I \neq \varnothing$
12: return $F_0$ and $F_1$
```

Algorithm 3: Recursive_simplicial_fixing (G)

some vertices. Line 11 ensures the recursive algorithm continues to run until G' has no more simplicial vertices.

The following proposition shows that the proposed algorithm runs in polytime.

Proposition 2 *Algorithm* 3 *runs in time* $O(mn^2)$.

Proof Lines 5 and 6, which find simplicial vertices and a maximum independent set of them, run in $O(mn)$ on graph G by Proposition 1.

Furthermore, the "while" loop will be repeated for at most n iterations.

So, the total run time is $O(mn^2)$. □

We also note that the complexity of the algorithm is $O(mn)$ if we run it only once on the input graph G. In Sect. 3, we run experiments for both single-time simplicial and recursive simplicial fixings. We also need to prove that the simplicial variable fixing is safe (i.e., the fixing procedure does not remove all optimal solutions) for any graph.

Theorem 1 *Let D be an independent set for simplicials of graph G. Then, there is an optimal solution $\tilde{x}$ with $\tilde{x}_v = 1$ for every vertex $v \in D$.*

Proof By the contradiction. Suppose there is a vertex $u \in D$ with $x^*_u = 0$ for any optimal solution x^*. We are to show that in every optimal solution x^*, there is exactly one neighbor $u' \in N_G(u)$ with $x^*_{u'} = 1$. Suppose not. Consider an optimal solution $\bar{x}$ with $\bar{x}_u = 0$ and $\bar{x}_{u'} = 0$ for every $u' \in N_G(u)$. Then, there is a solution $\hat{x}$ with $\hat{x}_u := 1$ and $\hat{x}_v := \bar{x}_v$ for every vertex $v \in V \setminus \{u\}$. However, this contradicts the optimality of $\bar{x}$. So, there is exactly one[1] neighbor $u' \in N_G(u)$ with $x^*_{u'} = 1$ for any optimal solution x^*.

[1] We note that at most one neighbor of u can be selected in an independent set as $G[N_G(u)]$ forms a clique in G.

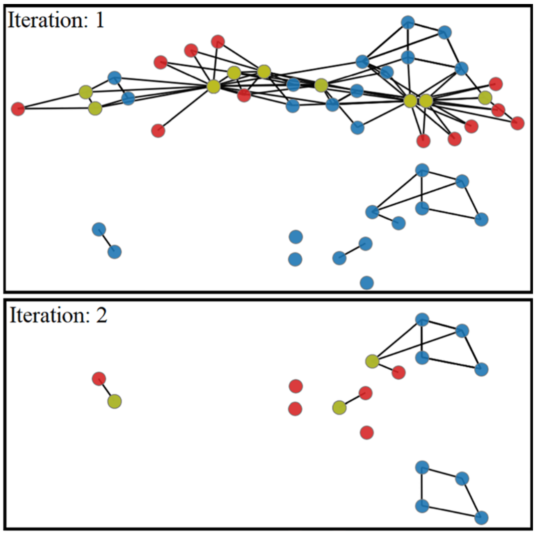

Fig. 2 Two iterations of the recursive simplicial fixing Algorithm 3 applied on the `karate` instance: red vertices denote a maximum independent set of simplicials that are fixed to one in the MIP model (1); yellow vertices show their corresponding neighbors that are fixed to zero in the MIP model (1); and blue vertices are the rest. The MIP model (1) need to solve only a subgraph of the instance with four vertices and edges at the end of the second iteration

Now consider an optimal solution $\bar{x}$ with $\bar{x}_u = 0$ and $\bar{x}_{u'} = 1$ for exactly one $u' \in N_G(u)$. Then, there is a feasible solution $\tilde{x}$ with $\tilde{x}_u = 1$ and $\tilde{x}_{u'} = 0$ for every $u' \in N_G(u)$. However, this solution has the same objective value as $\bar{x}$. This is a contradiction. □

We conclude this section with an illustration of Algorithm 3 on the `karate` graph. Figure 2 shows that two iterations of Algorithm 3 reduce the number of vertices and number of edges of the instance from 34 to four and 78 to four, respectively. Thus, it suffices to solve the MIP model (1) on a subgraph of the instance with only four vertices (corresponds to four decision variables) and four edges (corresponds to four constraints).

3 Computational experiments

This section reports our computational results on five sets of experiments:

(i) MIP model (1),
(ii) MIP model (1) preprocessed by a single round of Algorithm 3,
(iii) MIP model (1) preprocessed by recursive rounds of Algorithm 3,
(iv) MIP model (1) preprocessed by the LP-based fixing procedure of Nemhauser and Trotte [15], and
(v) MIP model (1) preprocessed by the LP-based fixing procedure of Nemhauser and Trotte [15] and then recursive rounds of Algorithm 3.

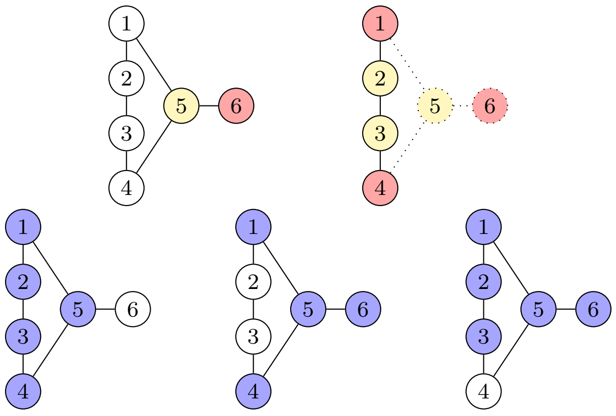

Fig. 3 (Top): Two iterations of our algorithm fix the pink and yellow vertices to one and zero, respectively. (Bottom): The induced subgraph of colored vertices is not chordal (left), has a claw (middle), and has a path P_5 (right)

We run our computational experiments on a set of existing social network instances. All of our codes are written in Python 3.8.12, and the MIP formulation (1) is solved by Gurobi 10.0.0. We also set a time limit of 3 h (10,800 s) for the Gurobi MIP solver. Furthermore, we ran our computational experiments on a Red Hat Enterprise Linux Workstation x64 version 7.6 with an Intel(R) Core(TM) i7-9800X CPU (3.8Ghz, 19.25MB, 165W) using 1 core with 32GB RAM. Our codes, data, and detailed results are available at: https://github.com/samuel-kroger/A-simplicial-fixing-for-the-maximum-stable-set-problem.

Table 1 shows our computational results for a subset of benchmark instances solved to optimality within the MIP time limit of 3 h. Among solved instances, we observe that the recursive simplicial fixing procedure fixes at least 94% of variables for all instances except for the `facebook` instance. Furthermore, the recursive approach results in the least total time among all solved instances except for the `facebook` instance. Another interesting observation is that `CA-CondMat` is solved to optimality by *only* the recursive simplicial fixing procedure.

Motivated by the fact that our fixing procedure solves the `CA-CondMat` instance to optimality, Fig. 3 provides a minimal instance that is also solved to optimality in the preprocess. Similar to `CA-CondMat`, this instance is also neither chordal, claw-free, P_5-free, nor perfect.[2] This motivates a future research direction to find new solvable classes of graphs for the MIS problem.

Table 2 summarizes our computational results for a subset of benchmark instances that are not solved to optimality within the MIP time limit of 3 h. Although none of these instances are solved within the time limit, we observe that the recursive

[2] It is not perfect because the clique number of the graph, which is 2, does not equal the chromatic number of the graph, which is 3.

Table 1 Computational results for benchmark instances that are solved within 3 h

Instance	*n*	*m*	Preprocess	Iters. (#)	Fixed (%)	Fix. time (s)	Total time (s)
			None	0	0.00	0.00	37.45
			One step	1	7.38	0.04	**33.05**
Facebook	4039	88,234	Recursive	6	**25.33**	0.07	35.36
			LP	1	6.07	0.00	38.58
			LP + recursive	7	**25.33**	0.04	35.49
			None	0	0.00	0.00	1.41
			One step	1	35.10	0.03	1.17
Wiki-Vote	7117	100,763	Recursive	18	99.94	0.07	0.07
			LP	1	**100.00**	0.00	**0.03**
			LP + recursive	2	**100.00**	0.00	**0.03**
			None	0	0.00	0.00	1.06
			One step	1	36.12	0.14	1.07
CA-CondMat	23,133	93,439	Recursive	5	**100.00**	0.15	0.15
			LP	1	38.22	0.00	0.70
			LP + recursive	5	**100.00**	0.09	**0.11**
			None	0	0.00	0.00	3.05
			One step	1	42.31	0.22	2.93
Brightkite	58,228	214,078	Recursive	13	99.78	0.35	0.36
			LP	1	96.03	0.00	0.17
			LP + recursive	10	**99.89**	0.02	**0.12**
			None	0	0.00	0.00	19.17
			One step	1	33.55	0.71	14.16
Gowalla	196,591	950,327	Recursive	27	99.31	1.53	1.83
			LP	1	85.60	0.00	1.88
			LP + recursive	26	**99.38**	0.24	**0.98**
			None	0	0.00	0.00	11.76
			One step	1	40.12	1.91	11.66
ca-citeseer	227,320	814,134	Recursive	5	99.99	1.87	1.87
			LP	1	49.10	0.00	6.02
			LP + recursive	5	**100.00**	1.06	**1.28**
			None	0	0.00	0.00	16.76
			One step	1	43.63	2.38	16.21
com-dblp	317,080	1,049,866	Recursive	6	**99.99**	2.40	2.41
			LP	1	58.86	0.00	5.94
			LP + recursive	7	**99.99**	0.96	**1.38**
			None	0	0.00	0.00	101.18

Table 1 (continued)

Instance	*n*	*m*	Preprocess	Iters. (#)	Fixed (%)	Fix. time (s)	Total time (s)
			One step	1	35.84	6.58	77.12
web-Google	875,713	4,322,051	Recursive	24	94.54	11.75	15.71
			LP	1	75.43	0.00	18.72
			LP + recursive	17	**98.25**	2.65	**5.30**
			None	0	0.00	0.00	49.73
			One step	1	57.00	5.58	47.77
com-youtube	1,134,890	2,987,624	Recursive	9	99.94	7.73	7.79
			LP	1	99.19	0.00	**2.17**
			LP + recursive	6	**99.99**	0.13	**2.17**

The bold and underlined numbers denote the best performances

Columns “iters. (#)”, “fixed (%)” and “fix. time” denote the number of preprocess iterations, the percentage of variables fixed to zero and one, and the fixing time (in seconds), respectively

Table 2 Computational results for benchmark instances that are not solved within 3 h

Instance	*n*	*m*	Preprocess	Iters. (#)	Fixed (%)	Fix. time (s)	Gap (%)
			none	0	0.00	0.00	9.72
			one step	1	3.30	0.03	8.06
socfb-Syracuse56	13,653	543,982	recursive	5	**9.32**	0.17	**6.54**
			LP	1	3.90	0.00	9.35
			LP + recursive	5	**9.32**	0.13	8.21
			none	0	0.00	0.00	5.59
			one step	1	3.80	0.03	5.53
socfb-Northeastern19	13,882	381,934	recursive	4	**10.51**	0.12	**5.44**
			LP	1	5.18	0.00	5.53
			LP + recursive	5	**10.51**	0.10	**5.44**
			none	0	0.00	0.00	14.93
			one step	1	0.00	0.13	14.93
smallworld	100,002	499,999	recursive	2	0.00	0.26	14.93
			LP	1	0.00	0.00	14.93
			LP + recursive	2	0.00	0.13	14.93
			none	0	0.00	0.00	0.26
			one step	1	0.54	0.51	0.26
flickrEdges	105,938	2,316,948	recursive	7	6.61	2.17	**0.23**
			LP	1	32.50	0.00	0.25
			LP + recursive	56	**57.87**	5.78	0.24

The bold and underlined numbers denote the best performances

Columns “iters. (#)”, “fixed (%)”, “fix. time”, and “gap (%)” denote the number of preprocess iterations, percentage of variables fixed to zero and one, the fixing time (in seconds), and the optimality gap percentage after 3 h, respectively simplicial fixing procedure reduced the optimality gap of `socfb-Syracuse56` by at least three percent.

We conclude this section with a comparison of our recursive approach with (i) the LP-based fixing procedure of Nemhauser and Trotte [15] (LP) and (ii) the LP-based fixing followed by our recursive approach (LP+recursive). In Table 1, our computational experiments show that the "LP+recursive" approach returns the most number of fixings for all instances. Furthermore, Table 1 shows the superiority of our recursive approach over the LP-based fixing procedure, except for the `Wiki-Vote` instance. In Table 2, we observe the superiority of the "LP+recursive" approach again. Furthermore, our recursive approach works at least as well as the LP-based fixing approach, except for the `flickrEdges` instance. However, the recursive approach returns a better optimality gap for the `flickrEdges` instance after 3 h.

4 Conclusion and future work

This paper proposes a recursive simplicial fixing procedure as a preprocessing algorithm for solving the maximum independent set problem. We prove that our proposed fixing procedure is safe and does not remove all optimal solutions from the feasible solution space. Our computational results show the effectiveness of the fixing procedure on a set of social network instances. We see the following future directions for this work: (i) understanding the underlying structure of the problem instances which yield good performance for the recursive fixing procedure, (ii) finding a broader class of graphs for which the MIS is polytime solvable, and (iii) developing similar fixing algorithms for other combinatorial optimization problems.

Acknowledgements This material is partially based upon work supported by Rice University's Building Research on Inequality and Diversity to Grow Equity (BRIDGE) seed grant. The authors thank the anonymous reviewer for helpful comments.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bondy, J., Murty, U.: Graph theory (2008)
2. Buchanan, A., Walteros, J.L., Butenko, S., Pardalos, P.M.: Solving maximum clique in sparse graphs: an $O(nm + n2^{d/4})$ algorithm for d-degenerate graphs. Optim. Lett. **8**, 1611–1617 (2014)
3. Buss, J.F., Goldsmith, J.: Nondeterminism within P^*. SIAM J. Comput. **22**(3), 560–572 (1993)
4. Butenko, S., Trukhanov, S.: Using critical sets to solve the maximum independent set problem. Oper. Res. Lett. **35**(4), 519–524 (2007)
5. Faenza, Y., Oriolo, G., Stauffer, G.: Solving the weighted stable set problem in claw-free graphs via decomposition. J. ACM **61**(4), 1–41 (2014)
6. Frank, A.: Some polynomial algorithms for certain graphs and hypergraphs (1976)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
8. Grötschel, M., Lovász, L., Schrijver, A.: 9.4 coloring perfect graphs. Geometric Algorithms and Combinatorial Optimization pp. 296–298 (1988)

9. Hammer, P.L., Hansen, P., Simeone, B.: Vertices belonging to all or to no maximum stable sets of a graph. SIAM J. Algebraic Discrete Methods **3**(4), 511–522 (1982)
10. Li, W., Zhu, B.: A $2k$-kernelization algorithm for vertex cover based on crown decomposition. Theoret. Comput. Sci. **739**, 80–85 (2018)
11. Lokshantov, D., Vatshelle, M., Villanger, Y.: Independent set in p 5-free graphs in polynomial time. In: Proceedings of The Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 570–581. SIAM (2014)
12. Mannino, C., Oriolo, G., Ricci, F., Chandran, S.: The stable set problem and the thinness of a graph. Oper. Res. Lett. **35**(1), 1–9 (2007)
13. Minty, G.J.: On maximal independent sets of vertices in claw-free graphs. J. Comb. Theory Ser. B **28**(3), 284–304 (1980)
14. Nakamura, D., Tamura, A.: A revision of Minty's algorithm for finding a maximum weight stable set of a claw-free graph. J. Oper. Res. Soc. Jpn. **44**(2), 194–204 (2001)
15. Nemhauser, G.L., Trotter, L.E., Jr.: Vertex packings: structural properties and algorithms. Math. Program. **8**(1), 232–248 (1975)
16. Nobili, P., Sassano, A.: An $O(n^2 \log n)$ algorithm for the weighted stable set problem in claw-free graphs. arXiv preprint arXiv:1501.05775 (2015)
17. Robson, J.M.: Finding a maximum independent set in time $O(2^{n/4})$. Tech. rep., Technical Report 1251-01, LaBRI, Université Bordeaux I (2001)
18. Salemi, H., Buchanan, A.: Solving the distance-based critical node problem. INFORMS J. Comput. **34**(3), 1309–1326 (2022)
19. Sbihi, N.: Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. Discret. Math. **29**(1), 53–76 (1980)
20. Walteros, J.L., Buchanan, A.: Why is maximum clique often easy in practice? Oper. Res. **68**(6), 1866–1895 (2020)