Using GRASP for the cover by s-defective independent sets problem

Austin BUCHANAN [a,1], Nannan CHEN [a] and Xin MA [a]

[a] *Department of Industrial and Systems Engineering, Texas A&M University 3131 TAMU, College Station, TX 77843-3131, USA*

Abstract. This paper considers the cover by s-defective independent sets problem (or, simply, the s-defective coloring problem), which generalizes the classical coloring problem. In the coloring problem, each color class should induce an edgeless subgraph. The s-defective coloring problem relaxes this requirement, allowing for at most s edges to appear in each color class's induced subgraph. We propose a local search neighborhood and apply the GRASP metaheuristic to the related minimization problem. The quality of the heuristically-generated solutions is evaluated with a commercial MIP solver. The integer programming formulation that we use generalizes the asymmetric representatives formulation.

Keywords. coloring, defective coloring, GRASP, heuristic

1. Introduction and background

This paper considers the cover by s-defective independent sets problem (or, simply, the s-defective coloring problem), which generalizes the classical coloring problem. In the coloring problem, each color class should induce an edgeless subgraph. The s-defective coloring problem relaxes this requirement, allowing for at most s edges to appear in each color class's induced subgraph. This is a generalization of coloring, since setting $s = 0$ yields the standard coloring definition.

Definition 1 (s-defective coloring)**.** *Given a graph $G = (V, E)$ and integer $s \geq 0$, an s-defective coloring of G is a partition $C_1, \ldots, C_k$ of V such that, for each $i = 1, \ldots, k$, the induced subgraph of C_i has at most s edges. We will call C_i an s-defective independent set.*

Analogously to the coloring problem, the s-defective coloring problem asks: find an s-defective coloring using the minimum number of partitions. The coloring problem is well-known to be computationally intractable. The problem of testing if a graph can be properly k-colored was one of Richard Karp's original 21 NP-complete problems [9]. Later it was shown that NP-completeness holds for any fixed $k \geq 3$ by the independent proofs of [11,17]. In contrast, 1-colorability is trivial to determine, and breadth-first search determines 2-colorability in linear

[1] Email: buchanan@tamu.edu

time. The problem of approximating the chromatic number is also hard – finding an $(n^{1-\epsilon})$-approximate solution is NP-hard for any constant $\epsilon > 0$ [4,18] – but giving each vertex its own color class gives an n-approximation. A naive algorithm determines if an n-vertex graph can be k-colored in time $O^*(k^n)$. The runtime has been significantly improved over the years to $O^*(2^n)$ by [1], and this algorithm actually calculates the chromatic number. The chromatic number can be found in polynomial time in the class of perfect graphs using semidefinite programming and the Lovász theta function [7].

Coloring has applications in scheduling and timetabling, register allocation, frequency assignment, and other fields [12]. The s-defective coloring problem has similar applications where the number of available colors is not sufficient, and the aim is to find a small, approximate coloring. Due to the intractibility of coloring, many heuristics have been proposed. See any of the numerous survey articles [15,5,12] for more information about coloring and heuristic approaches. It has been shown that the chromatic number is at most the degeneracy of a graph plus one, and a coloring achieving this bound can be found in linear time [13]. This upper bound based on degeneracy happens to be very good for many real-life instances. A more complicated construction heuristic called DSATUR, developed by David Brélaz, is another notable heuristic [2]. Perhaps the most effective coloring heuristic from the 2nd DIMACS implementation challenge was Craig Morgenstern's *Distributed coloration neighborhood search* [14]. Morgenstern's approach relied on the simulated annealing metaheuristic and an *impassé* neighborhood for local search. Another metaheuristic, GRASP, has been used for coloring [10] and related frequency assignment problems in mobile phone networks [6].

In this paper, we extend some of these well-known techniques for coloring to the generalized case of s-defective coloring. In Section 2, we provide a greedy construction heuristic. In Section 3, we describe a local search procedure. This approach is extended to use the GRASP metaheuristic in Section 4. Computational results are presented in Section 5 that compare the heuristics with a commercial MIP solver. The formulation used in this section generalizes the *asymmetric representatives formulation* [3] using formulation ideas from [16] to ensure that each color class induces a subgraph with at most s edges. Conclusions follow in Section 6.

2. Construction heuristic

We provide a simple, greedy construction heuristic for the s-defective coloring problem. We cannot expect to find an approximation with a nontrivial performance guarantee in arbitrary graphs, since finding an $(n^{1-\epsilon})$-approximate solution for any constant $\epsilon > 0$ is NP-hard.

Data: A graph $G = (V, E)$ and integer $s \geq 0$.
Result: An s-defective coloring $(C_1, \dots, C_k)$ of G.
initialize $V' \leftarrow V,\ k \leftarrow 0$;
while $V' \neq \emptyset$ **do**
    $k \leftarrow k + 1$;
    find a greedy maximal s-defective independent set $C_k \subseteq V'$ in $G[V']$;
    $V' \leftarrow V' \setminus C_k$;
end
return $(C_1, \dots, C_k)$;

Algorithm 1: Greedy heuristic for finding an s-defective coloring

3. Local search

We describe a neighborhood for a local search procedure in which a single vertex can be moved from one partition to another. Consider a graph $G = (V, E)$. Given a partition $C = (C_1, \dots, C_k)$ of V, we define the neighborhood of C as

$$N(C) := \{S = (S_1, \dots, S_k) : S \text{ is a partition of } V, \text{ and } |S_1 \Delta C_1| + \cdots + |S_k \Delta C_k| = 2\},$$

where Δ denotes the symmetric difference (i.e., $A \Delta B := (A \cup B) \setminus (A \cap B)$). We also define a penalty function

$$\Phi_s(C) := \sum_{i=1}^{k} (|E(G[C_i])| - s)^+ \tag{1}$$

which quantifies how far the partition C strays from being an s-defective coloring. A local search procedure can then be described as follows.

Data: A graph $G = (V, E)$, integer $s \geq 0$, and partition $C = (C_1, \dots, C_k)$ of V.
Result: A partition of V that is hopefully an s-defective coloring.
while $\exists S \in N(C)$ *with* $\Phi_s(S) < \Phi_s(C)$ **do**
    $C \leftarrow S$;
end
return $(C_1, \dots, C_k)$;

Algorithm 2: Local search for s-defective coloring

4. Metaheuristic – GRASP

We propose a greedy randomized construction heuristic for finding a partition of V that hopefully resembles an s-defective coloring. (Algorithm 3). The target k represents the desired number of partitions. The parameter α controls the tradeoff between greediness and randomness, where $\alpha = 0$ is purely greedy and $\alpha = 1$ is pure randomness. Pseudocode for the GRASP approach is found in Algorithm 4.

Data: A graph $G = (V, E)$, integer $s \geq 0$, minmax parameter $\alpha \in [0, 1]$, and target k.
Result: A partition $(C_1, \ldots, C_k)$ of V that hopefully resembles an s-defective coloring.
initialize $V' \leftarrow V,\ C_i \leftarrow \emptyset$ for $i = 1, \ldots, k$;
while $V' \neq \emptyset$ **do**
  pick a vertex $v \in V'$;
  $U \leftarrow \max\{|N(v) \cap C_i| : i = 1, \ldots, k\}$;
  $L \leftarrow \min\{|N(v) \cap C_i| : i = 1, \ldots, k\}$;
  $RCL \leftarrow \{i : i = 1, \ldots, k$ and $|N(v) \cap C_i| \leq L + \alpha(U - L)\}$;
  pick $i \in RCL$ uniformly at random;
  $C_i \leftarrow C_i \cup \{v\}$;
end
return $(C_1, \ldots, C_k)$;

Algorithm 3: Greedy randomized construction for s-defective coloring

Now we describe a GRASP for s-defective coloring. We start off by finding an upper bound on the s-defective chromatic number using Algorithm 1.

Data: A graph $G = (V, E)$, integer $s \geq 0$, and greed/random parameter α.
Result: A partition of V that is hopefully an s-defective coloring.
let C^* be s-defective coloring from Algorithm 1;
let k be number of partitions of C^*;
for $j = 1, \ldots, MaxIter$ **do**
  $C \leftarrow$GreedyRandomizedConstruction$(G, s, \alpha, k - 1)$;
  $C' \leftarrow$LocalSearch(C);
  if C' *is an s-defective coloring* **then**
    $C^* \leftarrow C'$;
    $k \leftarrow k - 1$;
  end
end
return C^*;

Algorithm 4: GRASP for s-defective coloring

In Algorithm 4 the procedure GreedyRandomizedConstruction is Algorithm 3. The procedure LocalSearch is Algorithm 2.

5. Computational results

In this section we provide an IP formulation for s-defective coloring and compare our GRASP results with the commercial MIP solver CPLEX.

5.1. IP Formulation

To avoid formulation symmetry, we generalize the *asymmetric representatives* formulation [3]. In this formulation, each color class has a representative vertex. The binary variable x_{ij} takes a value of one iff vertex i 'chooses' vertex j to be its representative. Note that a vertex can choose itself by setting $x_{ii} = 1$. The asymmetry occurs because vertices are only allowed to choose a representative with a lower index (assume the vertices are numbered $1, \ldots, n$). The binary variable z^j_{pq} takes a value of one iff adjacent vertices p and q are both assigned to color j. For a positive integer k, let $[k] := \{1, \ldots, k\}$.

$$\min \sum_{i \in [n]} x_{ii} \tag{2}$$

$$\sum_{j \in [i]} x_{ij} = 1,\ i \in [n] \tag{3}$$

$$x_{ij} \leq x_{jj},\ i > j \tag{4}$$

$$\sum_{\{p,q\} \in E\ :\ p > q \geq j} z^j_{pq} \leq s,\ j \in [n] \tag{5}$$

$$z^j_{pq} \geq x_{pj} + x_{qj} - 1,\ \{p,q\} \in E,\ p > q \geq j \tag{6}$$

$$x_{ij} \in \{0,1\},\ i,j \in [n] \tag{7}$$

$$z^j_{pq} \in \{0,1\},\ \{p,q\} \in E,\ p > q \geq j \tag{8}$$

Constraint (3) ensures that each vertex must choose one representative. Constraint (4) ensures that a vertex that is not selected as a representative (i.e., $x_{jj} = 0$) cannot be chosen to represent another vertex. Constraint (5) ensures that the color class represented by vertex j has at most s edges in its induced subgraph. Constraint (6) ensures that the edge $\{p,q\}$ will be counted in the color class represented by vertex j when adjacent vertices p and q both select j as representative.

5.2. Computational setup and instances used

The considered instances are the coloring instances from the 2nd DIMACS Implementation Challenge [8]. The GRASP computational experiments were conducted on a *Dell Precision WorkStation T7500*® computer, with eight 2.40 GHz Intel Xeon® processors, and 12 GB RAM. However, only one processor was used. The GRASP procedures were coded in C++ in the Microsoft Visual Studio 2010 environment. To solve the IP formulations, we used ILOG CPLEX 12.1® on a server with 8 GB RAM and four 3.33 GHz processors. Due to intractability of the problem, only twelve instances (three graphs and four values of s) were attempted to be solved using CPLEX.

5.3. Results

We first compare the GRASP approach with CPLEX on small instances (Table 1). Comprehensive results are provided for the GRASP approach (Table 2). Notice in Table 1 that the GRASP heuristic was able to quickly identify good solutions. In fact, CPLEX was not able to beat our heuristically found solutions in under an hour. However, this does not make for an entirely fair comparison, partially because we did not identify the time at which CPLEX found a solution of similar quality. On the other hand, CPLEX used four processors, while the GRASP implementation used one.

Table 1.: Computational experiments for GRASP (Algorithm 4) with $\alpha = 0.25$ and $MaxIter = n$ as compared with CPLEX. For each value of s, we report the time in seconds and the best solution found. A time limit of 3600 seconds was imposed.

		CPLEX		GRASP	
Graph	s	sol	time	sol	time
1-FullIns_3	0	4	0.87	4	0.01
1-FullIns_3	1	3	1.40	3	0.02
1-FullIns_3	2	3	0.96	3	0.01
1-FullIns_3	3	3	1.58	3	0.01
1-Insertions_4	0	5	>3600	5	0.03
1-Insertions_4	1	4	>3600	4	0.02
1-Insertions_4	2	4	>3600	4	0.02
1-Insertions_4	3	3	6.41	3	0.02
huck	0	11	7.38	11	0.04
huck	1	8	>3600	8	0.03
huck	2	7	>3600	7	0.03
huck	3	6	>3600	6	0.03

Table 2.: Computational experiments for GRASP (Algorithm 4) with $\alpha = 0.25$ and $MaxIter = n$. The parameters n and m denote the number of vertices and edges, respectively.

			$s = 0$		$s = 1$		$s = 2$		$s = 3$	
Graph	n	m	sol	time	sol	time	sol	time	sol	time
1-FullIns_3	30	100	4	0.01	3	0.02	3	0.01	3	0.01
1-FullIns_4	93	593	5	0.09	4	0.07	4	0.07	4	0.06
1-FullIns_5	282	3247	7	2.07	5	1.87	5	1.83	5	1.77
1-Insertions_4	67	232	5	0.03	4	0.02	4	0.02	3	0.02
1-Insertions_5	202	1227	6	0.55	5	0.51	5	0.51	4	0.45
1-Insertions_6	607	6337	7	13.22	6	12.97	6	12.87	6	12.86
2-FullIns_3	52	201	5	0.02	4	0.01	3	0.01	3	0.01
2-FullIns_4	212	1621	6	0.71	5	0.70	4	0.65	4	0.68
2-FullIns_5	852	12201	10	41.39	7	40.01	6	40.48	5	41.55
2-Insertions_3	37	72	4	0.00	3	0.00	3	0.00	3	0.00
2-Insertions_4	149	541	5	0.19	4	0.14	4	0.15	4	0.15
2-Insertions_5	597	3936	6	9.21	5	8.91	5	9.22	5	9.25
3-FullIns_3	80	346	6	0.05	4	0.04	4	0.04	3	0.03
3-FullIns_4	405	3524	8	4.30	6	3.91	5	4.21	4	4.18
3-FullIns_5	2030	33751	11	483.66	8	463.09	7	474.58	6	500.36
3-Insertions_3	56	110	4	0.01	3	0.01	3	0.01	3	0.01
3-Insertions_4	281	1046	5	0.83	4	0.75	4	0.81	4	0.78
3-Insertions_5	1406	9695	6	97.93	6	91.71	5	97.24	5	100.00
4-FullIns_3	114	541	7	0.11	5	0.10	4	0.09	4	0.09
4-FullIns_4	690	6650	11	19.70	6	17.62	6	17.79	5	19.07

4-Insertions_3	79	156	4	0.03	3	0.01	3	0.01	3	0.01
4-Insertions_4	475	1795	5	3.43	5	3.23	4	3.28	4	3.33
5-FullIns_3	154	792	8	0.26	5	0.21	5	0.21	4	0.21
5-FullIns_4	1085	11395	13	63.51	7	63.23	6	65.81	6	65.90
abb313GPIA	1557	53356	13	232.33	13	232.97	13	233.78	12	236.32
anna	138	493	11	0.17	8	0.14	7	0.13	6	0.12
ash331GPIA	662	4181	6	8.61	5	9.17	5	9.13	5	9.22
ash608GPIA	1216	7844	6	45.04	6	45.81	5	49.54	5	49.89
ash958GPIA	1916	12506	6	162.94	6	165.29	6	164.82	6	164.94
david	87	406	11	0.09	8	0.06	7	0.06	7	0.06
DSJC1000.1	1000	49629	33	240.36	31	243.15	28	251.22	27	250.00
DSJC1000.5	1000	249826	131	1685.50	125	1554.64	116	1503.97	110	1449.35
DSJC125.1	125	736	7	0.17	6	0.16	6	0.16	6	0.16
DSJC125.5	125	3891	25	0.77	20	0.73	19	0.70	18	0.69
DSJC125.9	125	6961	56	1.68	48	1.47	39	1.34	36	1.22
DSJC250.1	250	3218	12	2.21	11	2.19	10	2.21	10	2.17
DSJC250.5	250	15668	42	9.48	37	8.88	34	8.68	32	8.41
DSJC250.9	250	27897	100	24.01	91	21.07	78	19.40	72	17.98
DSJC500.1	500	12458	20	23.46	18	23.25	16	23.44	16	23.48
DSJC500.5	500	62624	75	126.56	68	114.70	63	108.05	59	106.10
DSJC500.9	500	112437	182	357.61	173	314.74	151	308.98	144	271.41
DSJR500.1	500	3555	14	7.78	13	7.68	12	7.31	11	7.28
DSJR500.1c	500	121275	105	315.72	229	397.93	157	404.20	155	342.94
DSJR500.5	500	58862	162	144.49	120	126.22	105	119.97	93	115.77
fpsol2.i.1	496	11654	66	18.11	42	15.97	39	14.86	32	14.72
fpsol2.i.2	451	8691	34	9.51	25	8.76	22	8.32	20	8.30
fpsol2.i.3	425	8688	34	8.80	25	8.21	22	7.81	20	7.42
games120	120	638	9	0.16	8	0.15	7	0.14	7	0.13
homer	561	1628	14	5.43	11	4.74	10	4.61	9	4.48
huck	74	301	11	0.04	8	0.03	7	0.03	6	0.03
inithx.i.1	864	18707	55	59.37	39	50.89	34	48.12	32	49.20
inithx.i.2	645	13979	32	23.17	25	21.16	20	20.17	20	19.57
inithx.i.3	621	13969	31	21.55	24	20.02	20	19.14	20	19.19
jean	80	254	10	0.04	7	0.03	6	0.03	6	0.03
le450_15a	450	8168	22	13.64	19	13.83	18	13.51	17	13.67
le450_15b	450	8169	21	13.70	20	13.61	18	13.61	17	13.50
le450_15c	450	16680	32	23.56	28	23.33	26	23.06	24	23.33
le450_15d	450	16750	32	24.02	29	23.71	26	23.42	25	23.31
le450_25a	450	8260	29	14.01	26	13.50	23	13.18	22	13.08
le450_25b	450	8263	28	13.98	25	13.71	23	13.28	22	13.33
le450_25c	450	17343	38	24.58	33	23.95	30	23.85	29	23.74
le450_25d	450	17425	38	24.80	34	23.98	31	24.00	29	23.83
le450_5a	450	5714	13	9.89	12	9.82	11	9.64	10	9.47
le450_5b	450	5734	13	9.94	12	9.72	11	9.66	10	9.43
le450_5c	450	9803	16	16.07	14	16.54	11	17.98	7	17.64
le450_5d	450	9757	16	16.31	14	16.45	7	17.85	7	17.88
miles1000	128	3216	44	0.77	30	0.69	26	0.63	23	0.59
miles1500	128	5198	74	1.21	43	1.07	39	0.98	32	0.91
miles250	128	387	8	0.11	7	0.10	6	0.10	6	0.10
miles500	128	1170	21	0.29	15	0.28	12	0.27	11	0.26
miles750	128	2113	34	0.48	22	0.43	20	0.43	17	0.43
mug100_1	100	166	4	0.03	4	0.03	4	0.03	3	0.03
mug100_25	100	166	4	0.03	4	0.03	4	0.03	3	0.03
mug88_1	88	146	4	0.02	4	0.02	4	0.02	3	0.02
mug88_25	88	146	4	0.02	4	0.02	4	0.02	3	0.02
mulsol.i.1	197	3925	49	1.70	31	1.41	26	1.33	24	1.28

mulsol.i.2	188	3885	31	1.22	23	1.16	18	1.08	18	1.00
mulsol.i.3	184	3916	31	1.22	23	1.10	18	1.00	18	0.99
mulsol.i.4	185	3946	31	1.26	23	1.16	18	1.01	18	0.98
mulsol.i.5	186	3973	31	1.25	23	1.18	18	1.02	18	1.04
myciel3	11	20	4	0.00	3	0.00	2	0.00	2	0.00
myciel4	23	71	5	0.00	4	0.00	3	0.00	3	0.00
myciel5	47	236	6	0.01	5	0.01	4	0.01	4	0.01
myciel6	95	755	7	0.10	6	0.09	5	0.09	5	0.08
myciel7	191	2360	8	0.83	7	0.79	7	0.80	6	0.78
qg.order30	900	26100	32	109.39	33	104.75	32	104.99	31	103.77
qg.order40	1600	62400	45	582.16	44	584.52	43	585.22	42	591.62
queen10_10	100	1470	14	0.23	13	0.22	12	0.21	11	0.19
queen11_11	121	1980	16	0.42	14	0.39	13	0.37	12	0.38
queen12_12	144	2596	16	0.71	16	0.66	15	0.62	13	0.64
queen13_13	169	3328	18	1.15	17	1.10	15	1.10	15	1.08
queen14_14	196	4186	20	1.82	18	1.75	17	1.69	16	1.67
queen15_15	225	5180	21	2.75	20	2.71	18	2.63	17	2.57
queen16_16	256	6320	23	4.10	21	4.01	19	3.96	18	3.85
queen5_5	25	160	5	0.00	6	0.00	5	0.00	5	0.00
queen6_6	36	290	8	0.01	7	0.01	6	0.01	6	0.01
queen7_7	49	476	9	0.02	9	0.02	8	0.02	8	0.02
queen8_12	96	1368	14	0.19	13	0.19	11	0.18	11	0.17
queen8_8	64	728	11	0.06	10	0.05	9	0.05	9	0.05
queen9_9	81	1056	13	0.11	11	0.11	10	0.11	10	0.10
school1	385	19095	43	21.99	38	21.28	37	20.12	32	20.02
school1_nsh	352	14612	39	15.19	34	14.37	31	13.71	29	13.63
wap05a	905	43081	59	188.60	54	184.58	49	182.13	46	179.42
wap06a	947	43571	59	206.78	53	202.78	48	198.79	45	195.45
will199GPIA	701	6772	10	17.49	9	17.49	8	17.64	8	17.55
zeroin.i.1	211	4100	50	1.80	30	1.64	27	1.38	23	1.43
zeroin.i.2	211	3541	30	1.22	21	1.16	19	1.12	17	1.05
zeroin.i.3	206	3540	30	1.19	22	1.16	19	1.11	17	1.10

6. Conclusion

We have applied GRASP metaheuristic for the s-defective coloring problem. For the small graphs that were solved by CPLEX in an hour, the approach finds optimal solutions. More experiments could be done to evaluate alternative neighborhood structures (e.g., mimicking Morgenstern's impassé neighborhood) or to find the 'best' values of the parameters α and $MaxIter$.

References

[1] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.

[2] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[3] M. Campêlo, V.A. Campos, and R.C. Corrêa. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156(7):1097–1111, 2008.

[4] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Computational Complexity, 1996. Proceedings., Eleventh Annual IEEE Conference on*, pages 278–287. IEEE, 1996.

[5] P. Galinier and A. Hertz. A survey of local search methods for graph coloring. *Computers & Operations Research*, 33(9):2547–2562, 2006.

[6] F.C. Gomes, P.M. Pardalos, C.S. Oliveira, and M.G.C. Resende. Reactive GRASP with path relinking for channel assignment in mobile phone networks. In *Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 60–67. ACM, 2001.

[7] M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. *Ann. Discrete Math*, 21:325–356, 1984.

[8] D.S. Johnson and M.A. Trick. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*, volume 26 of *Discrete Mathematics and Theoretical Computer Science*. AMS, 1996.

[9] R.M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[10] M. Laguna and R. Martí. A GRASP for coloring sparse graphs. *Computational optimization and applications*, 19(2):165–178, 2001.

[11] L. Lovász. Coverings and colorings of hypergraphs. In *Proc. 4th Southeastern Conference on Combinatorics, Graph Theory, and Computing, Utilitas Mathematica Publishing, Winnipeg*, pages 3–12, 1973.

[12] E. Malaguti and P. Toth. A survey on vertex coloring problems. *International Transactions in Operational Research*, 17(1):1–34, 2010.

[13] D.W. Matula and L.L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM*, 30(3):417–427, 1983.

[14] C. Morgenstern. Distributed coloration neighborhood search. In D.S. Johnson and M.A. Trick, editors, *Cliques, Coloring, and Satisfiability*, volume 26 of *Discrete Mathematics and Theoretical Computer Science*. AMS, 1996.

[15] P.M. Pardalos, T. Mavridou, and J. Xue. The graph coloring problem: A bibliographic survey. *Handbook of combinatorial optimization*, 2:331–395, 1998.

[16] H.D. Sherali and J.C. Smith. A polyhedral study of the generalized vertex packing problem. *Mathematical programming*, 107(3):367–390, 2006.

[17] L. Stockmeyer. Planar 3-colorability is polynomial complete. *ACM Sigact News*, 5(3):19–25, 1973.

[18] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2006.