Discrete Applied Mathematics 6 (1983) 243-254
North-Holland

EFFICIENT BOUNDS FOR THE STABLE SET, VERTEX COVER AND SET PACKING PROBLEMS*

Dorit S. HOCHBAUM
School of Business Administration, University of California, Berkeley, CA 94720, USA

Received 21 May 1982

In this paper we describe a collection of efficient algorithms that deliver approximate solution to the weighted stable set, vertex cover and set packing problems. All algorithms guarantee bounds on the ratio of the heuristic solution to the optimal solution.

1. Introduction

A *cover* in a graph G is a set of vertices C such that each edge of G has at least one endpoint in C. The *vertex cover problem* is the problem of finding a cover of the smallest weight in a graph whose vertices carry positive weights. This problem is known to be NP-complete even when the input is restricted to planar cubic graphs with unit weights [8]. A *stable set* in a graph is a set of pairwise nonadjacent vertices. The largest weight stable set is the complement of the smallest weight cover. Another related problem is the *set packing problem,* maximize wx subject to $Ax \le e$, x binary, e a column vector of ones and A a zero-one matrix. This problem can be represented as the stable set problem by constructing a graph (called the derived graph) whose vertices are columns of A, and two such vertices being adjacent if the two columns have a nonzero dot product.

We propose several heuristics to address the three problems above. All the heuristics are executed with a preprocessing step first. The preprocessing that relies on the concept of 'fixing variables' reduces the gap between the heuristic solution and the optimal solution. All the heuristics guarantee a solution to the vertex cover problem that is less than twice the weight of an optimal cover.

The algorithms proposed compare favorably with the 'greedy' heuristic [5] and the 'LP-heuristic' [10] designed to solve the set covering problem that generalizes the vertex cover problem. The weight of the cover delivered by the greedy heuristic is at most $\sum_{i=1}^{k} 1/i$ (k = the largest vertex degree) times the weight of the optimal cover. The 'LP-heuristic' cover weight equals at most twice the weight of an optimal cover. A special case of the 'LP-heuristic', the maximum matching heuristic [10], is not comparable to the algorithms proposed as the size of the cover delivered is at most $2-2/(r+1)$ times the size of the optimal cover, where r is the length of the largest odd cycle in graphs with all unit weights. Finding the parameter r is however

*This research was supported in part by the National Science Foundation under grant ECS-8204695.

an NP-complete problem, in that sense r is not a 'good' parameter. All of these factors are tight, i.e. there are graphs for which the weight of the heuristic cover is equal to the factor indicated times the weight of the optimal cover.

Section 3 discusses algorithms that color the graph. Given a weighted graph G with n vertices and maximum vertex degree k one algorithm takes only $O(kn^2 \log n)$ steps to deliver a cover whose weight is at most $2-2/k$ times the weight of an optimal cover, and a stable set of weight at least $2/k$ times the weight of an optimal stable set. The corresponding factor for the set packing problem depends on the maximum degree in the derived graph. The graph is constructed in number of steps proportional to the number of nonzero entries of the matrix A. Let $d(G)$ denote the maximum of the minimum degree of vertex in all subgraphs of G (this parameter can be evaluated quickly). The second 'coloration' algorithm delivers a cover whose weight is at most $2-2/(d(G)+1)$ times the weight of an optimal cover. For the stable set delivered, its weight is at least $2/(d(G)+1)$ the weight of an optimal stable set. The 'coloration' heuristic yields a particularly good approximate solution for graphs that are planar. The weight of the cover delivered is at most $\frac{3}{2}$ times the weight on an optimal cover, and the weight of the stable set at least $\frac{1}{2}$ the weight of an optimal stable set.

The algorithm described in Section 4 is especially useful for the set packing problem. It can be applied directly to the set packing problem without evaluating the derived graph first. The algorithm delivers a set packing whose weight is at least $1/p$ times the weight of the optimal solution, where p stands for the maximum column sum in A. The algorithm takes only $O(n \log n + pn)$ steps where n is the dimension of the vector x. Applying the algorithm to a weighted graph, it delivers a cover whose weight is at most $2-1/p$ the weight of an optimal cover, where p denotes the largest $p(G)+1$ such that G contains an induced $p(G)$-claw (a tree with $p(G)+1$ vertices, $p(G)$ of which are of degree one). The corresponding factor for the stable set delivered is $1/p$, delivering such a stable set takes only $O(n \log n + m)$ steps where m is the number of edges in the graph.

Section 5 describes an algorithm for graphs with unit weights. For graphs with *average degree* equal k the stable set delivered is of size at least $2/(k+1)$ the size of an optimal stable set. The size of the cover delivered is at most $2-2/(k_H+1)$ the size of an optimal cover, where k_H is the average degree in a certain subgraph H of G. These solutions are derived in only $O(kn^{3/2})$ steps.

2. Preprocessing the input

Many heuristics for the vertex cover problem may fare quite badly compared to the optimal solution to the problem. For instance, consider the renowned 'greedy' heuristic that finds a cover C in a weighted graph G:

Step 0: Set $C=\emptyset$.

Step 1: If G has no edges then stop, otherwise choose a vertex i minimizing the ratio between the vertex weight w_i and the number of neighbors of i.
Step 2: Add i to C, delete i and all edges incident with i from G and return to Step 1.

As shown by Johnson [12], this heuristic applied to a graph of maximum degree k, may deliver a cover whose weight exceeds the weight of an optimal cover by a factor of $\sum_{i=1}^{k} 1/i$ even if all weights are unit.

Another example is available when the set of vertices of the graph V is split into stable sets $V_1, V_3, \ldots, V_k$. (Methods of obtaining such a split will be described in Section 3.) Each set $V \setminus V_i$ is a cover. In particular $V \setminus V_i$ of the smallest weight may seem to be a good candidate for a cover C. Still, the ratio $w(C)/w(C^*)$ may be arbitrarily large even when G is fixed. (Here and elsewhere, $w(S)$ stands for the weight of a set S of vertices). Indeed, consider the path with vertices $1,2,3,4$ and weights $w_1 = w_4 = M$, $w_2 = w_3 = 1$ for some large M. When V is partitioned into two stable sets, the strategy proposed here yields a cover C with $w(C) = M+1$ and yet the optimal cover C^* has $w(C^*) = 2$. Some more illustrations of such undesirable behavior of other heuristics will be discussed in detail later.

Nevertheless, the quality of the solution delivered by any heuristic can be improved if we first partition the graph into two subgraphs with the property that in one subgraph an optimal selection of a cover is known and in the other the weight of the optimal cover is at least half of the total weight of all vertices. The existence of such a partition, implied by the fractional solution to the problem, has been established by Nemhauser and Trotter [4]. They suggested to solve first the 'LP relaxation',

$$\begin{array}{lll} \text{minimize} & \sum w_i x_i, & \\ \text{subject to} & x_i + x_j \geq 1 & \text{for all edges } ij, \\ & x_i \geq 0 & \text{for all vertices } i, \end{array}$$

and then use a trick known as 'fixing variables'. In at least one optimal solution to the relation, each x_i has value $0, \frac{1}{2}$, or 1. If we write

$$\begin{array}{ll} i \in P & \text{if } x_i = 1, \\ i \in Q & \text{if } x_i = \frac{1}{2}, \\ i \in R & \text{if } x_i = 0, \end{array}$$

then

(i) at least one optimal cover in G contains P,
(ii) each vertex in R has all its neighbors in P,
(iii) each cover in G has weight at least $w(P) + \frac{1}{2} w(Q)$.

From (i) and (ii), it follows instantly that at least one optimal cover in G consists of the set P and of an optimal cover in the subgraph H induced by Q. Thus it suffices to find an optimal cover in H; working with H rather than with G is what is meant by 'fixing variables'.

Fixing variables is a trick which can be applied not only in the context of finding optimal covers but also in the context of heuristics for finding near-optimal covers. In this new context, the trick has a nice corollary: If C is any cover in H, then (by (ii)) $P \cup C$ is a cover in G and (by (iii)) its weight is at most twice the weight of an optimal cover. Thus any heuristic for finding near-optimal covers can be made to deliver a cover whose weight is at most twice the weight of an optimal cover: it suffices to preprocess G by finding P, Q, R and then to apply the heuristic to H rather than directly to G. Can the upper bound of two be replaced by a smaller constant? We suspect that the answer is negative even in the special case of unit weights.

Conjecture. Unless $P = NP$, there is no polynomial-time algorithm which, given an arbitrary graph, delivers a cover whose size is at most C times the size of an optimal cover for some constant C smaller than two.

The power of preprocessing can be demonstrated on the 'greedy' heuristic mentioned earlier. If the graph is preprocessed and the heuristic applied to H then the weight of the resulting cover in G is at most twice the weight of an optimal cover, where without preprocessing it could be arbitrarily large. Incidentally, the greedy heuristic with preprocessing does not constitute a counterexample to the conjecture made above. To justify this claim, consider the graph G arising from $k+1$ disjoint cliques ($k \geq 2$), each of them having k vertices, and introduce k^2 additional vertices, each of which is joined to the original $k(k+1)$ vertices. It is an easy exercise to show that (a) the preprocessing step yields $H = G$, (b) the greedy heuristic applied to G delivers a cover of size $k^2 + (k+1)(k-1)$ and (c) the smallest cover in G has size $k(k+1)$. As k increases, the ratio $(2k^2-1)/(k^2+k)$ gets arbitrarily close to two.

As suggested by Edmonds and Pulleybank, and noted in [14], the LP relaxation can be solved by finding an optimal cover C in the bipartite graph with two vertices a_i, b_i of weight w_i for each vertex i of G, and two edges $a_i b_j, a_j b_i$ for each edge ij of G: then it suffices to set

$$\begin{aligned} x_i &= 1 \quad \text{if } a_i, b_i \in C, \\ x_i &= \tfrac{1}{2} \quad \text{if } a_i \in C,\ b_i \notin C \text{ or } a_i \notin C,\ b_i \in C, \\ x_i &= 0 \quad \text{if } a_i \notin C,\ b_i \notin C. \end{aligned}$$

In turn, the problem of finding C can be reduced into a minimum cut problem: the bipartite graph can be converted into a network by making each edge $a_i b_i$ into a directed arc $a_i b_j$ of an infinite capacity, adding a source s with an arc sa_i of capacity w_i for each i and adding a sink t with an arc $b_i t$ of capacity w_i for each i. Now a minimum cut S, T with $s \in S$ and $t \in T$ points out the desired C: it suffices to set $a_i \in C$ iff $a_i \in T$ and $b_i \in C$ iff $b_i \in S$. The minimum cut can be found by algorithms of Galil [7] and Sleator [15] in only $O(n^{5/3} m^{2/3})$ and $O(nm \log n)$ steps, respectively, with n standing for the number of vertices in G and m standing for the number of edges in G. In short, it takes only $O(nm \log n)$ steps to preprocess G by partitioning the set of its vertices into P, Q and R.

3. Easily colorable graphs

Theorem 1. *Let G be a weighted graph with n vertices and m edges; let k be an integer greater than one. If it takes only s steps to color the vertices of G in k colors (so that adjacent vertices have distinct colors), then it takes only $s+\mathrm{O}(nm\log n)$ steps to find a stable set whose weight is at least $2/k$ times the weight of an optimal stable set and to find a cover whose weight is at most $2-2/k$ times the weight of an optimal cover.*

Proof. It takes only $s+\mathrm{O}(nm\log n)$ steps to color G in k colors and to find the set P, Q, R of the preceding section. (Note that it suffices to color only the vertices of Q.) The coloring of G splits Q into k color classes; if S denotes the heaviest of them then $w(S)\geq w(Q)/k$. The set $R\cup S$ is stable; since

$$w(R\cup S)\geq w(R)+\frac{1}{k}\,w(Q)\geq\frac{2}{k}\,(w(R)+\frac{1}{2}\,w(Q)),$$

its weight is at least $2/k$ times the weight of an optimal stable set. The complement C of $R\cup S$ is a cover; since

$$w(C)\leq w(P)+\frac{k-1}{k}\,w(Q)\leq\frac{2(k-1)}{k}\,(w(P)+\frac{1}{2}\,w(Q)),$$

its weight is at most $2-2/k$ times the weight of an optimal cover. □

The remainder of this section consists of various corollaries of Theorem 1. To begin with, let $d(G)$ denote the largest d such that G contains a subgraph in which each vertex has degree at least d. As proved by Szekeres and Wilf [16], every graph G can be colored in $d(G)+1$ colors. For the sake of completeness, we shall describe a way of finding such a coloring and evaluating $d(G)$ in only $\mathrm{O}(n+m)$ steps. To evaluate $d(G)$, it suffices to dismantle G by successive removals of vertices of minimum degree.

Step 0: Set $d=0$.
Step 1: If G has no vertices left then stop; otherwise choose a vertex v of the smallest degree.
Step 2: Replace d by the maximum of d and the degree of v. Then remove v (and all the edges incident with v) from G and return to Step 1.

If v_i denotes the vertex removed from G in the ith iteration then each v_i has at most d neighbors among the vertices $u_{i+1}, v_{i+2}, \ldots, v_n$. To color G in no more than $d+1$ colors, it suffices to scan the sequence of v_i's from v_n to v_1, assigning to each v_i the smallest positive integer not yet assigned to any of its neighbors.

Corollary 1A. *It takes only $\mathrm{O}(nm\log n)$ steps to find, in any weighted graph G*

with n vertices and m edges such that $m>0$, a stable set whose weight is at least $2/(d(G)+1)$ times the weight of an optimal stable set and a cover whose weight is at most $2-2/(d(G)+1)$ times the weight of an optimal cover.

The celebrated theorem of Brooks [4] asserts the following: if G is a connected graph of a maximum degree k such that $k\geq 3$ and if G is not the complete graph with $k+1$ vertices then G is k-colorable. An elegant and constructive proof of this theorem, due to Lovász [13], provides an algorithm which finds the coloring in only $O(kn)$ steps. (The algorithm requires finding cutpoints and endblocks in a graph. This can be done in $O(m)$ steps by depth-first search as described, for instance, in [3].)

Corollary 1B. *It takes only $O(kn^2 \log n)$ steps to find, in any weighted graph with n vertices and a maximum degree k such that $k\geq 2$, a stable set whose weight is at least $2/k$ times the weight of an optimal stable set and a cover whose weight is at most $2-2/k$ times the weight of an optimal cover.*

Proof. We may assume that $k\geq 3$: otherwise each component is a cycle or a path and a straightforward dynamic programming algorithm finds an optimal stable set and an optimal cover in only $O(n)$ steps. Furthermore, we may assume that the graph is connected: otherwise each component may be treated separately. Finally, we may assume that the graph is not complete: otherwise an optimal stable set and an optimal cover may be found trivally in $O(n)$ steps. But then the desired conclusion follows directly from Brooks' theorem and Theorem 1. □

The coloration heuristics are not counterexamples to our conjecture. To see that we let a graph G be defined as follows. Consider k k-cliques and k $(k-1)$-stable sets. Each clique has one edge connecting it to one of the vertices of a stable set. $k-1$ of the stable sets are one set of vertices in a complete bipartite graph with the kth stable set as the second set of vertices. For such family of graphs, one can easily verify that G is k chromatic and $G=H$. One feasible[1] k-coloration consists of each one of the stable sets colored by one of the k-colors. The heuristic then delivers a cover C of size $(2k-1)(k-1)$. the Optimum cover C^* is of size $k(k-1)+(k-1)$ and the ratio

$$w(C)/w(C^*)=2-\frac{2-3/k}{k-1/k}$$

which could be arbitrarily close to 2.

The standard proof due to Heawood that every planar graph is five-colorable

[1] In order to make this coloration unique we add few edges to the graph: The vertices in the cliques that connect each clique to each stable set are linked together to make a complete subgraph. The ith stable set is connected to all these vertices except for the ith vertex.

(see, for instance, [9]) is easiiy convertible into an algorithm which actually finds the coloring in only $O(n^2)$ steps[2].

Corollary 1C. *It takes only* $O(n^2 \log n)$ *steps to find, in any weighted planar graph with n vertices, a stable set whose weight is at least* 0.4 *times the weight of an optimal stable set and a cover whose weight is at most* 1.6 *times the weight of an optimal cover.*

Furthermore, the proof that every planar graph is four-colorable [1,2] is convertible into an algorithm which actually finds the coloring in only a polynomial number of steps.

Corollary 1D. *It takes only a polynomial number of steps to find, in any weighted planar graph, a stable set whose weight is at least* 0.5 *times the weight of an optimal stable set and a cover whose weight is at most* 1.5 *times the weight of an optimal cover.*

4. Set-packing problems

By a *p-claw* in a graph, we shall mean distinct vertices, $u_0, u_1, \dots, u_p$ such that u_0 is adjacent to the p vertices $u_1 u_2, \dots, u_p$ but no two of the p vertices $u_1 . u_2, \dots, u_p$ are adjacent to each other.

Theorem 2. *It takes only* $O(n \log n + m)$ *steps to find, in any weighted graph G with n vertices, m edges and no* $(p+1)$*-claw, a stable set whose weight is at least* $1/p$ *times the weight of an optimal stable set.*

It takes only $O(nm \log n)$ *steps to find a cover in the graph G whose weight is at most* $2 - 1/p$ *times the weight of the optimal cover.*

Proof. The algorithm goes as follows.

Step 0: Sort the vertices in a sequence $1, 2, \dots, n$ such that $w_1 \geq w_2 \geq \cdots \geq w_n$. Set $S = \emptyset$.

Step 1: If G has no vertices then stop; otherwise take the smallest j which is a vertex of G and proceed to Step 2.

Step 2: Add j to S, delete j and all its neighbors (along with all the edges incident with at least one of these vertices) from G and return to Step 1.

[2] Recently linear algorithms were devised to 5-color a planar graph: (1) "A linear 5-coloring algorithm of planar graphs" by N. Chiba, T. Nishizeki and N. Saito, J. Algorithms 2 (1981) 317-327; (2) "Two linear-time algorithms for 5-coloring a planar graph" by D. Matula, Y. Shiloach and R. Tarjan, Stanford Department of Computer Science, Report No. STAN-CS-80-830.

Let v_i denote the vertex added to S in the ith iteration and let V_i denote the set of vertices deleted from G in the ith iteration. If S^* is a stable set then there are at most p vertices in each $S^* \cap V_i$ and the weight of each of them is at most the weight of w_i. Hence $w(S^* \cap V_i) \leq pw_i$ for all i, and so $w(S^*) \leq pw(S)$.

The ratio $1/p$ for the stable set cannot be improved in the worst case using preprocessing. For the vertex cover problem the ratio $[w(V)-w(S)]/[w(V)-w(S^*)]$ can be reduced from p without preprocessing to $2-1/p$ with preprocessing. We apply the algorithm for finding a stable set on the subgraph of G, H. The complement of the stable set is a cover of weight not exceeding $w(P)+w(Q)-w(S^*)/p$. Dividing by the weight of the optimal cover $w(P)+w(Q)-w(S^*)$, yields an upper bound on the ratio $2-1/p$. □

The following example illustrates that preprocessing does not necessarily increase the weight of the stable set delivered by the heuristic in the worst case, and it also proves that the bound for the cover problem could be arbitrarily close to 2 (with preprocessing). Consider a graph G consisting of a p-claw and a t-clique. Each vertex of the claw, except for the 'center', has unit weight. The 'center' of the claw has weight $1+\varepsilon$, and all the vertices of the clique have weight ε. t is chosen such that $t > (p-1)/\varepsilon - 1$. Each vertex of the claw is linked to all vertices of the clique. The algorithm in Theorem 2 delivers a stable set S of weight $1+\varepsilon$ and the weight of the optimal stable set is p. It is easy to check that $G=H$ hence the preprocessing does not improve the ratio $w(S)/w(S^*)$ in the worst case. Taking the complement of the stable set S yields a cover C of weight $p+\varepsilon t$ where the weight of an optimal cover C^* is $1+\varepsilon+\varepsilon t$. By the choice of t,

$$w(C)/w(C^*) \leq (2p-1)/(p+\varepsilon) = 2-(1+2\varepsilon)/(p+\varepsilon),$$

thus the ratio could be arbitrarily close to 2. For the vertex cover problem the preprocessing is essential. If preprocessing was not used then for a graph G that is a p-claw with center weighted $1+\varepsilon$ and all other vertices 1 the weight of the cover delivered by the heuristic without preprocessing is p where the weight of the optimum cover is $1+\varepsilon$. (For this particular graph applying preprocessing yields immediately the optimal solution.)

Note that the ratio $1/p$ featured in Theorem 2 is neither necessarily majorized nor necessarily minorized by the ratio $2/k$ featured in Corollary 1B: the largest p such that G contains a p-claw may be anywhere between one and the largest degree k. Although it is an NP-complete problem to decide whether G contains a p-claw or not, Theorem 2 has an immediate application to the set-packing problems maximize wx subject to $Ax \leq e$, x binary. The point is that the derived graph has no $(p+1)$-claw with p standing for the largest column sum in A. Note also that, in order to apply the algorithm of Theorem 2 to the derived graph G, one does not have to make an explicit list of all the edges in G. If A is represented as a sparse matrix by lists $R_1, R_2, \ldots, R_m$ and $C_1, C_2, \ldots, C_n$ such that each R_i lists all the columns j with $a_{ij}=1$ and each C_j lists all the rows i with $a_{ij}=1$, then the neighbor of a specified vertex of G can be easily detected and deleted: the time spent on all the executions of Step 2 is only proportional to the total number of nonzeros in A.

Corollary 2A. *It takes only* $O(n \log n + pn)$ *steps to find, in any set-packing problem whose matrix A has n columns and the largest column sum p, a feasible solution whose value is at least* $1/p$ *times the optimal value.*

5. Unweighted sparse graphs

Sparse graphs have large stable sets. More precisely, the celebrated Theorem of Turán [17] asserts that every graph with n vertices and an average degree k (this quantity is not necessarily an integer) contains a stable set of size at least $n/(k+1)$. An elegant proof of Turán's theorem, due to Erdös [6], is easily converted into the following algorithm for finding a stable set S in at most $O(m)$ steps.

Step 1: Set $S = \emptyset$.
Step 1: If G has no vertices then stop; otherwise choose a vertex v with the smallest degree.
Step 2: Add v to S, delete v and all its neighbors (along with all the edges incident with at least one of these vertices) from G and return to Step 1.

To show that the size $|S|$ of the stable set delivered upon termination is at least $n/(k+1)$, we observe that whenever a vertex v_i of degree d_i is chosen and deleted we eliminate a total of $d_i + 1$ vertices from the graph and at least $\frac{1}{2}d_i(d_i+1)$ edges. The minimum number q of vertex selections when

$$\sum_{i=1}^{q} d_i(d_i+1) \le nk \quad \text{and} \quad \sum_{i=1}^{q} (d_i+1) = n$$

is attained for $d_i + 1 = n/q$ for all i. Solving

$$q \cdot \frac{n}{p}\left(\frac{n}{q} - 1\right) \le nk$$

we find that $q \ge n/(k+1)$. If we apply the algorithm to the subgraph H induced by the set of vertices Q, the value of the stable set delivered is at least $2/(k_H+1)$ the size of a maximum stable set when k_H is the average degree in the graph H. Though k_H can be much larger compared to k, the average degree in G, we still have the following result.

Theorem 3. *In any graph G with n vertices and average degree k it takes* $O(kn^{3/2})$ *steps to find a stable set of size at least* $2/(k+1)$ *times the size of maximum stable set.*

Proof. Preprocessing an unweighted graph can be executed in only $O(m\sqrt{n})$ steps [11]. Once the partition P, Q, R is obtained we apply the algorithm above to the subgraph H. The total number of steps does not exceed $O(kn^{3/2})$. The size of the stable set delivered by the algorithm is at least $|R|+|Q|/(k_H+1)$ where k_H is the average degree in H. (Incidently, note that $k_H \geq 2$ as there is always a solution with no vertices of degree one in the subgraph H.) Now we note that:

Fact 1. G is a connected graph, hence

$$k \geq \frac{|Q|k_H+|R|+|P|}{|Q|+|R|+|P|}$$

(note that $n=|Q|+|R|+|P|$).

Fact 2. $|R| \geq |P|$, otherwise setting $G=H$ (i.e., all vertices are assigned the value $\frac{1}{2}$) implies an 'LP relaxation' solution of value larger than $|R|+\frac{1}{2}|Q|$, contradiction.

To complete the proof it suffices to show using Fact 1 that

$$\frac{|R|+|Q|/(k_H+1)}{|R|+\frac{1}{2}|Q|} \geq 2\left(\frac{|Q|k_H+|R|+|P|}{|Q|+|R|+|P|}+1\right)^{-1}.$$

Rearranging this inequality we reduce it to

$$|Q|(|R|k_H(k_H-1)-|P|(k_H-1)-|P|(k_H-1)) \geq 0.$$

The validity of this inequality follows easily from fact 2. □

Unlike the other algorithms discussed in previous sections we cannot show that the bound $1/(k+1)$ for the ratio $w(S)/w(S^*)$ is indeed tight when S is the set delivered by the algorithm without preprocessing. Nevertheless we know that this factor is strictly less than $2/(k+1)$. In that sense the algorithm in Theorem 3 (that includes preprocessing) does increase in the worst case the size of the stable set. An example that illustrates this fact is given by a graph on 50 vertices. The graph consists of 12 3-claws and a 2-clique. The two vertices of the 2-clique are linked to all 36 pendant vertices of the claws. The average degree of such a graph is 4.36, the size of the optimal stable set –36, and the algorithm without preprocessing selects (with ties broken arbitrarily) 13 vertices (12 claw centers and one vertex in the 2-clique). Now, $13/36<2/(4.36+1)$ hence the ratio $2/(k+1)$ cannot be guaranteed without preprocessing. Using preprocessing the algorithm delivers the optimal solution for this particular graph.

The complement of the stable set delivered by the algorithm of Theorem 3 is a cover C of size at most $2-2/(k_H+1)$ times the size of an optimal cover C^*. Unfortunately we cannot in general replace k_H by k in this factor. (To see why the stable set problem differs from the vertex cover problem in this case, consider any graph G augmented by a huge p-claw. The average degree in the augmented graph can be set to be arbitrarily close to 1 with increasing values of p. The sizes of the cover delivered by the algorithm C and the optimal cover C^* are increased by 1 compared to their sizes in the graph G where the optimal and algorithm stable sets are increased by p. That means that the ratio of the optimal and heuristic stable sets gets asymptotically close to 1 as the average degree gets close to 1, but the heuristic to optimal covers ratio remains fixed. If the heuristic to optimal covers ratio were bounded by $2-2/(k+1)$, then for k sufficiently close to 1 we could have derived the optimal cover in G using the heuristic. That of course is impossible unless $P=NP$.) One corollary of this observation is that the guaranteed upper bound on the ratio $|C|/|C^*|$ is at least $\frac{4}{3}$ since $k_H \geq 2$. Finally we describe a counter example to illustrate that this algorithm does not violate our conjecture. Consider t cycles of size 6 each all having the first and fourth vertices v_1, v_4 in common. Add to this graph one more cycle of size 7 with the same two vertices v_1, v_4 as first and fourth (or first and fifth depending which direction one counts). Let the set of all neighbours of the vertex v_1 be $N(v_1)$. The vertex v_5 adjacent to v_4 on the 7-cycle consists with the set $N(v_1)$ of a stable set. We introduce edges between each vertex of the set $N(v_1) \cup \{v_5\}$ and each vertex of $V \setminus N(v_1) \setminus \{v_1, v_4, v_5\}$, and all possible edges between all vertices of $V \setminus N(v_1) \setminus \{v_1, v_4, v_5\}$. The number of vertices in this graph is $4t+7$. It is easy to check that $G=H$ and that $N(v_1) \cup \{v_5\}$ is an optimal stable set of size $2t+3$. The algorithm selects (ties broken arbirarily) vertices v_1, v_4, v_6 in this order (v_6 is adjacent to v_5 on the 7-cycle), $|C| = 4t+4$ and $|C^*| = 2t+4$. The ratio $|C|/|C^*| = 2-2/(t+2)$ is arbitrarily close to 2 for graphs G with t arbitrarily large.

Acknowledgement

The work described in this paper was motivated by discussions with V. Chvátal. I wish to thank him for the generous help and the time and effort he devoted.

References

[1] K. Appel and W. Haken, Every planar map is four colorable. Part I: Discharging, Illinois J. Math. 21 (1977) 429-490.

[2] K. Appel, W. Haken and J. Koch, Every planar map is four colorable. Part II: Reducibility, Illinois J. Math. 21 (1977) 491-567.

[3] S. Baase, Computer Algorithms: Introduction to Design and Analysis (Addison-Wesley, Reading, MA, 1978).

[4] R.L. Brooks, On coloring the nodes of a network, Proc. Cambridge Philos. Soc. 37 (1941) 194-197.

[5] V. Chvátal, A greedy-heuristic for the set-covering problem, Math. Operations Research 4(3) (1979) 233-235.

[6] P. Erdös, On the graph-theorem of Turán, Math. Lapok 21 (1970) 249-251.

[7] Z. Galil, A new algorithm for the maximal flow problem, Proc. Nineteenth Annual Symposium on Foundations of Computer Science (1978) 231-245.

[8] M.R. Garey, D.S. Johnson and L. Stockmeyer, Some simplified NP-complete graph problems, Theoret. Comput. Sci. 1 (1976) 237-267.

[9] F. Harary, Graph Theory (Addison-Wesley, Reading, MA, 1969).

[10] D.S. Hochbaum, Approximation algorithms for the set covering and vertex cover problems, SIAM J. Comput. 11(3) (1982), also W.P. #64-79-80, GSIA, Garnegie-Mellon University, April 1980.
[11] J.E. Hopcroft and R.M. Karp, A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SIAM J. Comput. 2 (1973) 225-231.
[12] D.S. Johnson, Approximation algorithms for combinatorial problems, J. Comput. System Sci. 9 (1974) 256-278.
[13] L. Lovász, Three short proofs in graph theory, J. Combin. Theory (B) 19 (1975) 269-271.
[14] G.L. Nemhauser and L.E. Trotter, Vertex packings: structural properties and algorithms, Math. Programming 8 (1975) 232-248.
[15] D.D.K. Sleator, An $O(nm \log n)$ algorithm for maximum network flow, Doctoral Dissertation, Dept. Computer Science, Stanford University, 1980.
[16] G. Szekeres and H.S. Wilf, An inequality for the chromatic number of a graph, J. Combin. Theory 4 (1968) 1-3.
[17] P. Turán, An external problem in graph theory, Mat. Fiz. Lapok 48 (1941) 436-452.