A Combinatorial Algorithm for Computing a Maximum Independent Set in a t-perfect Graph

Friedrich Eisenbrand [*] *Stefan Funke* [*†] *Naveen Garg* [‡] *Jochen Könemann* [§]

Abstract

We present a combinatorial polynomial time algorithm to compute a maximum stable set of a t-perfect graph. The algorithm rests on an ε-approximation algorithm for general set covering and packing problems and is combinatorial in the sense that it does not use an explicit linear programming algorithm or methods from linear algebra or convex geometry. Instead our algorithm is based on basic arithmetic operations and comparisons of rational numbers which are of polynomial binary encoding size in the input.

1 Introduction

A *stable* or *independent* set $S \subseteq V$ of an undirected graph $G = (V,E)$ is a subset of the nodes of G which are pairwise nonadjacent. The *stable set problem* is the problem of finding a stable set of a graph G with maximum cardinality and is NP-hard [5].

The stable set problem can be formulated as an integer program with the following linear programming relaxation:

$$(1.1) \qquad \begin{array}{rll} & \max & \sum_{v \in V} y_v \\ \{u,v\} \in E: & y_u + y_v & \leq 1 \\ u \in V: & y_u & \geq 0. \end{array}$$

This linear programming relaxation (1.1) has fractional vertex solutions. An *odd cycle* of G is a multiset $C = \{v_1, \dots, v_k\}$ of nodes such that $k \leq |V|$ is odd and $\{v_i, v_{i+1}\} \in E$ for all $i = 1, \dots, k-1$ and $\{v_k, v_1\} \in E$ [1]. If C is an odd cycle, then each characteristic vector of a stable set of G satisfies the *odd cycle inequality*

$$(1.2) \qquad \sum_{v \in C} y_v \leq \lfloor |C|/2 \rfloor.$$

For a graph G we denote the polytope defined by the inequalities (1.1) and all odd cycle inequalities (1.2) by $P_{\mathrm{odd}}(G)$.

A graph G is *t-perfect*, if its odd cycle polytope P_{odd} is a 0/1 polytope. The study of these graphs was suggested by Chvátal [2]. The odd cycle inequalities can be *separated* in polynomial time [6], i.e., one can decide in polynomial time whether one of the inequalities (1.2) violates a given point x^* and if so, find such an inequality. From the *equivalence of separation and optimization* [10, 12, 8] it follows that the *ellipsoid method* [11] can be used to optimize a linear function over $P_{\mathrm{odd}}(G)$ in polynomial time and consequently solve the weighted stable set problem for t-perfect graphs in polynomial time. However, the ellipsoid method relies heavily on division, rounding and approximation. This is due to the fact that, conceptually, one even leaves the field of rationals and needs to compute square roots. This introduces the issue of necessary accuracy such that the algorithm is guaranteed to run in polynomial time and to produce correct results in the bit-model.

1.1 Main results. We provide the first combinatorial algorithm for computing a maximum stable set in a t-perfect graph. This we achieve by considering the dual of the linear programming relaxation of the integer program for the maximum stable set in t-perfect graphs. The dual, which is a covering problem, can be solved to within a $(1+\alpha)$ approximation using a combinatorial algorithm for approximating fractional set cover. For a suitable choice of α this leads to a procedure for computing the size of a maximum stable set in a t-perfect graph which is then used to construct the actual stable set.

We believe that such an approach could be applied to other combinatorial optimization problems for which the only algorithms known are based on the ellipsoid method. Further, while our algorithm for maximum stable set in t-perfect graphs does not significantly outperform the algorithm based on the ellipsoid method, it does provide a combinatorial approach to this problem, which we believe can be fine-tuned to obtain a much faster algorithm.

[*] `eisen, funke@mpi-sb.mpg.de`, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany

[†] Partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

[‡] `naveen@cse.iitd.ernet.in`, Indian Institute of Technology, New Delhi. This work was done while the author was visiting the Max-Planck-Institut für Informatik, Germany on a Humboldt award.

[§] `jochen@cmu.edu`, Carnegie Mellon University, GSIA 211 Posner Hall, Pittsburgh, PA 15213. This material is based upon work supported by the National Science Foundation under Grant No. 0105548.

[1] If C is just a set, we have a simple cycle; but here we are also allowing non-simple cycles.

1.2 Related work. Combinatorial algorithms for the stable set problem of t-perfect graphs are not known and a non-polyhedral characterization of the class of t-perfect graphs has, so far, not been provided. However some subclasses of t-perfect graphs have been identified which can be characterized and recognized in polynomial time. These include bipartite graphs, almost bipartite graphs [4], series parallel graphs [1, 2], graphs that do not contain an *odd* K_4 [6] and graphs that do not contain a so called *bad* K_4 [7]. Combinatorial algorithms for the stable set problem have been provided for bipartite graphs, almost bipartite graphs [4] and series parallel graphs [1, 2].

The search for combinatorial algorithms for polynomial problems for which only ellipsoidal algorithms are known has stimulated a lot of research in the field of combinatorial optimization. Recent breakthroughs include the combinatorial algorithms for submodular function minimization [14, 9] and the combinatorial algorithm for path-matching problems [15].

The combinatorial algorithm for approximating fractional set cover is an unpublished result of Garg and Könemann; both the algorithm and its analysis are a straightforward extension of the greedy algorithm for integral set cover [3]. Similar such algorithms have been obtained by Plotkin, Shmoys and Tardos [13] using Lagrange relaxations and by Young [17, 18] using the technique of oblivious rounding.

2 A $(1+\alpha)$-approximation algorithm for the fractional set cover problem

The *minimum weight set cover* problem is defined as follows. Given a universe $U = \{e_1, \dots, e_n\}$ of n elements, a (potentially exponential-size) collection $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ of subsets of U and a cost function $c : \mathcal{S} \to \mathbb{Q}^+$, find a subset of $\mathcal{S}$ of minimum cost which covers all the elements in U.

The problem can be formulated as an integer linear program with the following linear programming relaxation:

$$\begin{array}{lrll} (2.3) & \min & \sum_{S \in \mathcal{S}} c_S x_S & \\ & e \in U: & \sum_{S : e \in S} x_S & \geq 1, \\ & S \in \mathcal{S}: & x_S & \geq 0. \end{array}$$

In this section we will present a combinatorial $(1+\alpha)$-approximation algorithm for the linear programming relaxation of the set cover problem.

We will first review the well-known greedy algorithm which computes an integral set cover with cost at most $O(\log n)$ times the optimum, see, e.g. [16, p. 108]. For this we consider the dual of the set cover linear program (2.3):

$$\begin{array}{lrll} (2.4) & \max & \sum_{e \in U} y_e & \\ & S \in \mathcal{S}: & \sum_{e \in S} y_e & \leq c_S \\ & e \in U: & y_e & \geq 0. \end{array}$$

In each round, the greedy algorithm picks a subset $S \in \mathcal{S}$ which minimizes the ratio $\rho_S = c_S / (\sum_{e \in S} r_e)$, where r_e denotes the 'requirement' of element e, i.e. we have $r_e = 1$ if it is not covered by any set which we have picked so far, $r_e = 0$ otherwise. In the following we assume that we can find such a subset $S \in \mathcal{S}$ with minimum ρ_S value in polynomial time.

We set $r_e = 0$ for the newly covered elements $e \in U$ in this round and proceed until all $e \in U$ are covered. How can we bound the 'quality' of this solution?

Consider the dual of the set cover linear program (2.4). In each round, when picking a set S (and therefore setting $r_e = 0$ for all $e \in S$ which were not covered before), we set each dual variable y_e of the newly covered elements of this round to $y_e := \rho_S \cdot r_e$.(Here, of course, we have $r_e = 1$, but we will later change the algorithm such that values $r_e \notin \{0, 1\}$ might occur.)

Clearly, in any stage of this algorithm we have $\sum_{S \in \mathcal{S}} x_S c_S = \sum_{e \in U} y_e$, but only the primal solution is feasible at the end. How 'infeasible' can the dual solution be? Consider the constraint for one particular $S \in \mathcal{S}$ in the dual

$$\sum_{e \in S} y_e \leq c_S,$$

and let $e_1, \dots, e_m$ be an ordering of the elements of S in increasing order of the rounds in which the elements were covered. We claim that $y_{e_i} \leq c_S / (m - i + 1)$. Assume that S' is the set that first covers e_i. As S' is chosen in such a way that the average cost of the newly covered elements in S' is minimum, this average cost must be less than the average cost of the elements which would be newly covered by S. At this point in time, there are at least $m - i + 1$ uncovered elements of S and therefore $y_e \leq c_S / (m - i + 1)$. Hence we have

$$\sum_{e \in S} y_e \leq c_S \cdot \sum_{h=1}^{h=m} 1/h \leq c_S \cdot \ln(n+1).$$

So each dual constraint is violated by at most a factor of $\ln(n+1)$. Hence, scaling the dual variables by $1/\ln(n+1)$ yields a feasible dual solution which has objective value at least $1/\ln(n+1)$ times the value of the primal (feasible) solution.

PROPOSITION 2.1. *The greedy algorithm computes a set cover which has size at most* $\ln n$ *times the size of the optimal set cover.*

How can we modify the algorithm to obtain primal and dual solutions which are at most $(1+\alpha)$ away from each other? The crucial idea is not to select a set S 'entirely', i.e. setting $x_S = 1$, but just adding some small value ε to x_S, thereby obtaining a greedy algorithm with finer granularity. The choice of ε will depend on the quality of the approximation we wish to achieve.

So again, in each round we 'pick' the set S which minimizes the *cost-to-requirement ratio*

$$\rho_S = c_S / \sum_{e \in S} r_e$$

and update the primal variables

$$x_S = x_S + \varepsilon,$$

as well as the dual variables

$$y_e = y_e + \rho_S \cdot r_e \cdot \varepsilon/(1+\varepsilon), \qquad e \in S.$$

Furthermore we decrease the requirements

$$r_e = r_e/(1+\varepsilon), \qquad e \in S.$$

Notice here that the decrease Δ_{r_e} of the requirement of $e \in S$ is not ε, as in the simple greedy algorithm for set covering. Instead, the decrease is $\Delta_{r_e} = r_e \cdot \varepsilon/(1+\varepsilon)$. The dual variable y_e is basically increased by $\rho_S \Delta_{r_e}$. If at some point the requirement r_e falls below some threshold δ (which we chose later on), we set it to zero.

LEMMA 2.1. *At any time throughout the algorithm* $\sum_{S \in \mathcal{S}} c_S x_S = (1+\varepsilon) \sum_{e \in U} y_e$ *holds.*

Proof. In each round, if the primal objective function is increased by $c_S \cdot \varepsilon$, the dual objective function value is increased by $\sum_{e \in S} \rho_S \cdot r_e \cdot \varepsilon/(1+\varepsilon)$ and since $\rho_S = c_S/(\sum_{e \in S} r_e)$, the dual increase is $c_S \cdot \varepsilon/(1+\varepsilon)$.

When the algorithm terminates, we have for each $e \in U$, $\sum_{S: e \in S} x_S \geq \varepsilon \cdot k$, where k is the smallest number such that $(1+\varepsilon)^{-k} \leq \delta$. Scaling primal variables by $1/(\varepsilon \cdot k)$ will therefore ensure a primal feasible solution.

Let us now argue how far the dual solution is from feasibility.

LEMMA 2.2. *For any set* $S \in \mathcal{S}$, *we have* $\sum_{e \in S} y_e \leq c_S \ln((1+\varepsilon) \cdot n/\delta)$.

Proof. Consider the sum of the requirements $\sum_{e \in S} r_e$ of S. This quantity is $|S|$ at the beginning and at least δ in the 'last' round where some element of S is covered. Consider a round where some set S' has minimum ratio $\rho_{S'}$ and where the requirements of S are changed, i.e., there exists an element $e \in S \cap S'$ with $r_e > 0$.

If the sum of the requirements is decreased by some amount $\beta = \sum_{e \in S \cap S'} r_e \cdot (\varepsilon/(1+\varepsilon))$, then $\sum_{e \in S} y_e$ increases by $\rho_{S'} \cdot \beta$ which is at most $\rho_S \cdot \beta$.

Now consider the ratio $\rho_S = c_S / \sum_{e \in S} r_e$ as a function of the requirement $\sum_{e \in S} r_e$ of the elements in S. It is easy to see that the sum of all increments of $\sum_{e \in S} y_e$ is bounded by the area under the curve c_S/x for $\delta/(1+\varepsilon) \leq x \leq |S|$ which is $c_S \cdot \ln((1+\varepsilon)|S|/\delta) \leq c_S \ln((1+\varepsilon)n/\delta)$. Hence scaling the dual variables by $1/\ln((1+\varepsilon)n/\delta)$ yields a dual feasible solution.

The ratio between the scaled primal and dual solutions is now equal to $(1+\varepsilon)\ln((1+\varepsilon)n/\delta)/\varepsilon k$ which is bounded by $(1+\varepsilon)\ln((1+\varepsilon)n/\delta)/\varepsilon \log_{1+\varepsilon} \delta^{-1}$. Now we need to choose our constants so that the above quantity is no more than $(1+\alpha)$. Setting $\delta = ((1+\varepsilon)n)^{-1/\varepsilon}$ and substituting this in the above expressions gives

$$\frac{(1+\varepsilon)\ln((1+\varepsilon)n)^{1+1/\varepsilon}))}{\varepsilon \log_{1+\varepsilon}((1+\varepsilon)n)^{1/\varepsilon}}$$

$$= \frac{(1+\varepsilon)(1+1/\varepsilon)\ln(1+\varepsilon)n)}{\log_{1+\varepsilon}(1+\varepsilon)n}$$

$$= (1+\varepsilon) \cdot (1+\varepsilon) \cdot \frac{\log(1+\varepsilon)}{\varepsilon} \leq (1+\varepsilon)^2$$

So to obtain a ratio of $(1+\alpha)$, we have to set $\varepsilon = \sqrt[2]{1+\alpha} - 1$.

THEOREM 2.1. *Given a fractional set cover problem, one can compute primal and dual feasible solutions x and y to the covering linear program and its dual packing linear program respectively such that $c^T x / \mathbf{1}^T y \leq (1+\alpha)$ with $O(P(n) \cdot n \cdot \log n/\alpha^2)$ operations, where $P(n)$ denotes the time to determine the set with minimum cost-to-requirement ratio* $\min_{S \in \mathcal{S}} \rho_S$.

Proof. Each $e \in U$ needs $k = \log_{1+\varepsilon} \delta^{-1}$ rounds until its requirement drops below δ. So in the worst case we need $n \cdot \log_{1+\varepsilon} \delta^{-1}$ rounds, which for our choice of $\delta = n^{-\frac{1}{\varepsilon}}$ becomes $n \ln n/(\varepsilon \cdot \ln(1+\varepsilon))$. As for small ε we have $\ln(1+\varepsilon) \geq \varepsilon/2$, we obtain $O(n \cdot \log n/\varepsilon^2)$ rounds overall or $O(n \cdot \log n/\alpha^2)$ as for small α, we have $\sqrt[2]{1+\alpha} - 1 > \alpha/4$. Thus in the real RAM model, the actual running time is $O(P(n) \cdot n \cdot \log n/\alpha^2)$ where $P(n)$ denotes the time to determine the variable x_i to increase in each round.

3 Computing the size of a maximum independent set of a t-perfect graph

In the following we will use the approximation algorithm from the previous section to derive a combinatorial algorithm for *exactly* solving the linear programming relaxation of the independent set problem on t-perfect graphs.

Given a graph $G = (V,E)$ with vertex set V and edge set E, an *independent* or *stable set* is a subset $I \subseteq V$ such that all nodes in I are pairwise nonadjacent. The *maximum stable set problem* is the problem of finding a stable set of maximum cardinality.

Here we will apply the combinatorial approximation algorithm for the covering LP to the problem of finding a maximum stable set in a t-perfect graph. We believe that this outlines a general framework to obtain combinatorial algorithms for certain packing problems with exponentially many constraints like matching, path matching etc.

Our goal is to solve exactly the following linear program defined by a t-perfect graph via a combinatorial algorithm. Let $\mathcal{C}$ denote the set of odd cycles (not necessarily simple) of a graph G.

$$(3.5)\qquad \begin{array}{rll} \max & \sum_{v\in V} y_v & \\ \{u,v\}\in E: & y_u + y_v & \leq 1 \\ C \in \mathcal{C}: & \sum_{v\in C} y_v & \leq \lfloor |C|/2 \rfloor, \\ u \in V: & y_u & \geq 0. \end{array}$$

Observe that our independent set LP is in fact the dual of the primal set cover LP where we have 2-element sets S_e for each edge and a subset S_c for each odd cycle.

The rough idea will be as follows: As we know that $\sum_{v\in V} y_v \leq n$ holds for each 0-1 solution to (3.5), we know that in particular, the optimal solution has objective function value $\mathrm{OPT}_I S \leq n$. So choosing $\alpha = 1/n$, and running our $(1+\alpha)$ approximation algorithm will yield a feasible solution whose objective value is at least $\mathrm{OPT}/(1+\alpha) > \mathrm{OPT} - 1$. In other words we can determine the *exact* size of the maximum stable set of graph G. We will use this as a kind of counting oracle later on to actually *construct* a maximum stable set for G. Another, more geometric view of the outcome of the approximation algorithm is that we obtain a point inside the stable set polytope such that all vertices of the polytope, which have higher objective value indeed have the *optimal* objective value. This geometric interpretation can also be used to arrive at a maximum stable set of G, i.e. an optimal vertex of the polyhedron.

3.1 Using the $(1+\alpha)$ approximation algorithm for set cover

We will apply our $(1+\alpha)$ approximation algorithm for fractional set cover to the dual of our independent set LP with odd cycle constraints. Again, in the dual (the set cover LP), we have a variables x_e for each edge $e \in E$ and variables x_C for each odd cycle $C \in \mathcal{C}$. The dual linear program looks as follows:

$$(3.6)\qquad \begin{array}{rll} \min & \sum_{e\in E} x_e + \sum_{C\in\mathcal{C}} x_C \cdot \lfloor |C|/2 \rfloor, & \\ v \in V: & \sum_{e: v\in e} x_e + \sum_{v\in C} x_C & \geq 1, \\ e \in E: & x_e & \geq 0, \\ C \in \mathcal{C}: & x_C & \geq 0. \end{array}$$

Observe that there might be an exponential number of variables x_C in this linear program. But recall that the $(1+\alpha)$ approximation algorithm only has to determine in each round, which variable to increase by ε, and using the notation from Section 2 this was the set $S \in \mathcal{S}$ with minimum cost-to-requirement ratio ρ_S.

The minimum ratio $\rho_e, e \in E$ is easily determined for the variables x_e, but not as trivially determined for the x_C as there might be exponentially many of them.

We will now describe how to find the minimum ratio $\min_{C\in\mathcal{C}} \rho_C$, where

$$\rho_C = \lfloor |C|/2 \rfloor / \sum_{v\in C} r_v,$$

and $r_v \in [0,1]$ denotes the requirement of a vertex $v \in V$. As we only have to consider odd cycles up to a length n, we can assume, $|C|$ is known and try all possible odd lengths $|C| \leq n$. So given $|C|$ we want to find a (possible non-simple) odd cycle C' such that $|C| = |C'|$ and $\sum_{v\in C'} r_v$ is maximized.

This odd cycle C' can be easily computed by the following idea. We construct $|C|+1$ copies $V^1, V^2, \ldots, V^{|C|+1}$ of the vertex set V. Denote the i-th copy of a vertex v by $v^{(i)}$. For each original undirected edge $\{v,w\} \in E$, we draw a directed edge $e = (v^{(i)}, w^{(i+1)})$ for $i = 1, \ldots, |C|$ in this *layered graph*. The weight of such an edge will be $w(e) = 1 - r_w \geq 0$. This defines an acyclic graph G^* which has $(|C|+1)\cdot|V|$ nodes and $|C|\cdot|E|$ edges.

Then we compute for every node $v \in V$ the shortest path from its representative in V^1 to its representative in $V^{|C|+1}$. Clearly, this path has length $|C|$ and represents a (possibly non-simple) odd cycle in the original graph. On the other hand, each odd (possibly non-simple) cycle W in the original graph of length $|C|$ is represented as a path from $v^{(1)}$ to $v^{(|C|+1)}$, where v is a member of W. The length of this path in the layered graph is equal to $|C| - \sum_{v\in W} r_v$. So the shortest such path is the one which maximizes $\sum_{v\in W} r_v$ which is exactly what we need.

PROPOSITION 3.1. *Given an undirected graph $G = (V,E)$ and node weights $r_v \in [0,1]$ for $v \in V$, one can compute the minimum cost-to-requirement ratio $\rho_C = \lfloor |C|/2 \rfloor / \sum_{v\in C} r_v$ in time $O(V^2 \cdot E)$.*

Proof. It suffices to solve the single source shortest path problem in the layered graph with $|V|+1$ layers for each source node v in V^1 in time $O(V \cdot E)$ each. The length of all odd cycles involving v can be read off the respective representatives of v in the odd layers of the constructed graph in $O(V)$ time, which implies the assertion.

LEMMA 3.1. *The size of a maximum stable set of a t-perfect graph can be computed in $O(V^5 \cdot E \cdot \log V)$ arithmetic operations on numbers of length at most $O((n \log n)^2)$ bits.*

Proof. From the preceding discussion it follows that a $1 + 1/n$ approximation to the covering program (3.6) yields the size of the largest stable set in a t-perfect graph G. Combining Proposition 3.1 and Theorem 2.1 and choosing $\alpha = 1/n$ yields a bound of $O(V^5 \cdot E \cdot \log V)$ arithmetic operations.

A bound on the size of the numbers involved can be obtained by noting that the requirements on the vertices are of the form $(1+\varepsilon)^i/(1+\varepsilon)^k$ where $0 \leq i \leq k$ and $k = O(\varepsilon^{-2} \log n)$. Since, $\varepsilon = O(n^{-1})$ the requirements on the vertices can be represented using only $O(k \log n) = O((n \log n)^2)$ bits.

4 Constructing a maximum stable set

In this section we will show how to use our $(1+\alpha)$ approximation algorithm as a *counting oracle* to actually *construct* a maximum independent set of a graph G. For our algorithm we use the fact that t-perfectness is a hereditary property. This result is folklore but we provide a proof for the sake of completeness.

LEMMA 4.1. *Let $G = (V,E)$ be a t-perfect graph and $u \in V$ be a vertex of G. Then the graph G_u obtained by removing u from G is t-perfect.*

Proof. We have to show that the odd cycle polytope $P_{\text{odd}}(G_u)$ of G_u is integral. To see this, it is enough to show that $P_{\text{odd}}(G_u)$ is the projection of the face $F = P_{\text{odd}}(G) \cap \{x \in \mathbb{R}^n \mid x_u = 0\}$ onto the variables $x_v, v \in V - \{u\}$. We write a point x of F in the form $(0, x_{V-\{u\}})$, where $x_{V-\{u\}}$ are the components of x indexed by $V - \{u\}$. Notice that a point x of the form $(0, x_{V-\{u\}})$, where $x_{V-\{u\}} \in P_{\text{odd}}(G_u)$ cannot violate an odd cycle of G which uses the vertex u. From this it follows that $F \supseteq \{x \in \mathbb{R}^n \mid x = (0, x_{V-\{u\}}), x_{V-\{u\}} \in P_{\text{odd}}(G_u)\}$. Since an odd cycle of G_u is also an odd cycle of G, we conclude also that $F \subseteq \{x \in \mathbb{R}^n \mid x = (0, x_{V-\{u\}}), x_{V-\{u\}} \in P_{\text{odd}}(G_u)\}$. From this we conclude the lemma, since the face of an integral polyhedron is again an integral polyhedron.

The construction of a maximum stable set of a t-perfect graph G now works as follows. We iteratively construct independent sets $S_0 \subseteq S_1, \ldots, \subseteq S_k$ of G and graphs $G_0, \ldots, G_k$ such that S_k is a maximum independent set of G and G_{i+1} results from G_i via the deletion of one or several nodes. We initialize S with the empty set and maintain the following invariant:

There exists a maximum independent set of G which is the union of S_i and a maximum independent set of G_i.

We begin with $S_0 = \emptyset$ and $G_0 = G$. In step i, compute a $(1 + 1/n)$ approximation of the linear program (3.6) defined by G_i. This procedure gives us the size k_i of a maximum stable set of G_i. Now we check whether a particular vertex $u \in V_i$ is a member of all maximum stable sets of G_i by removing the vertex from G_i. The resulting graph $G_{i,u}$ is still t-perfect. We again run the $(1+1/n)$-approximation algorithm on the linear program (3.6) defined by $G_{i,u}$. If the size of a maximum independent set of $G_{i,u}$ is less than k_i, then u has to be in each maximum stable set of G_i. In this case we update $S = S + u$ and continue with the graph G_{i+1}, which results from G_i via removing u and all the neighbors of u. If the size of a maximum independent set of $G_{i,u}$ is k, then we continue the procedure with $G_{i+1} = G_{i,u}$. Clearly, the procedure terminates after n rounds with the correct result.

Using Lemma 3.1 we obtain the following running time for our procedure.

THEOREM 4.1. *There exists a combinatorial algorithm which computes a maximum stable set of a t-perfect graph in time $O(V^6 \cdot E \cdot \log V)$.*

Final remarks

We presented a combinatorial algorithm for the maximum stable set problem for t-perfect graphs, which does not make use of an explicit linear programming algorithm or geometrical tools such as the ellipsoid method. The crux of the algorithm is to make use of a general $(1+\alpha)$-approximation scheme for packing/covering problems which we presented in Sections 2 and 3. Making use of the structure of the stable set polytope, we employ this approximation scheme in Section 4 as a *counting oracle* to actually construct a maximum independent set of a t-perfect graph.

The problem of moving from an approximate frational stable set solution to an integral solution of at least the same quality can be phrased as the problem of moving from a (possibly interior) point of a polytope to a vertex solution with an at least as large objective function value. This can be done by purely geometric means. However we were aiming for a purely combinatorial algorithm and hence have not elaborated on this approach.

References

[1] M. Boulala and J.-P. Uhry. Polytope des indépendants d'un graphe série-parallèle. *Discrete Mathematics*, 27(3):225–243, 1979.

[2] V. Chvátal. On certain polytopes associated with graphs. *Journal of Combinatorial Theory Ser. B*, 18:138–154, 1975.

[3] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[4] J. Fonlupt and J.-P. Uhry. Transformations which preserve perfectness and H-perfectness of graphs. In *Bonn Workshop on Combinatorial Optimization (Bonn, 1980)*, pages 83–95. North-Holland, Amsterdam, 1982.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann, 1979.

[6] A. M. H. Gerards and A. Schrijver. Matrices with the Edmonds-Johnson property. *Combinatorica*, 6:365–379, 1986.

[7] A. M. H. Gerards and F. B. Shepherd. The graphs with all subgraphs t-perfect. *SIAM Journal on Discrete Mathematics*, 11(4):524–545 (electronic), 1998.

[8] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[9] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. In ACM, editor, *Proceedings of the 32nd annual ACM Symposium on Theory of Computing*, pages 97–106, New York, NY, USA, 2000. ACM Press.

[10] R. M. Karp and C. H. Papadimitriou. On linear characterizations of combinatorial optimization problems. *SIAM Journal on Computing*, 11(4):620–632, 1982.

[11] L.G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1097, 1979.

[12] M. W. Padberg and M. R. Rao. The russian method for linear programming III: Bounded integer programming. Technical Report 81-39, New York University, Graduate School of Business and Administration, 1981.

[13] S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, 20(2):257–301, 1995.

[14] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory. Series B*, 80(2):346–355, 2000.

[15] B. Spille and R. Weismantel. A generalization of Edmonds' matching and matroid intersection algorithms. In *Proceedings of the Ninth International Conference on Integer Programming and Combinatorial Optimization*, volume 2337, pages 9–20. Springer, 2002.

[16] V. V. Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2001.

[17] N. E. Young. Randomized rounding without solving the linear program. In *Proceedings of the 6th Annual Symposium on Discrete Algorithms*, pages 170–178. ACM Press, 1995.

[18] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 538–546, 2001.