Solving the Stable Set Problem in Terms of the Odd Cycle Packing Number

Adrian Bock[1], Yuri Faenza[1], Carsten Moldenhauer[1], and Andres Jacinto Ruiz-Vargas[2]

1 **DISOPT, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland**
2 **DCG, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland**

Abstract

The classic *stable set* problem asks to find a maximum cardinality set of pairwise non-adjacent vertices in an undirected graph G. This problem is NP-hard to approximate with factor $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$ [10, 24], where n is the number of vertices, and therefore there is no hope for good approximations in the general case. We study the stable set problem when restricted to graphs with bounded *odd cycle packing* number $\mathrm{ocp}(G)$, possibly by a function of n. This is the largest number of vertex-disjoint odd cycles in G. Equivalently, it is the logarithm of the largest absolute value of a sub-determinant of the edge-node incidence matrix A_G of G. Hence, if A_G is totally unimodular, then $\mathrm{ocp}(G) = 0$. Therefore, $\mathrm{ocp}(G)$ is a natural distance measure of A_G to the set of totally unimodular matrices on a scale from 1 to $n/3$.

When $\mathrm{ocp}(G) = 0$, the graph is bipartite and it is well known that stable set can be solved in polynomial time. Our results imply that the odd cycle packing number indeed strongly influences the approximability of stable set. More precisely, we obtain a polynomial-time approximation scheme for graphs with $\mathrm{ocp}(G) = o(n/\log n)$, and an α-approximation algorithm for any graph where α smoothly increases from a constant to n as $\mathrm{ocp}(G)$ grows from $O(n/\log n)$ to $n/3$. On the hardness side, we show that, assuming the exponential-time hypothesis, stable set cannot be solved in polynomial time if $\mathrm{ocp}(G) = \Omega(\log^{1+\varepsilon} n)$ for some $\varepsilon > 0$. Finally, we generalize a theorem by Györi *et al.* [9] and show that graphs without odd cycles of small weight can be made bipartite by removing a small number of vertices. This allows us to extend some of our above results to the *weighted* stable set problem.

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases stable set problem, independent set problem, approximation algorithms, odd cycle packing number, maximum subdeterminants

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2014.187

1 Introduction

The *stable* or *independent set* problem is fundamental in combinatorial optimization. It is as follows: Given an undirected graph $G = (V, E)$, find a subset $S \subseteq V$ of maximum cardinality such that no two vertices in S are adjacent. Stable sets occur naturally when pairwise conflicts of choice have to be respected. It is among the very first combinatorial optimization problems that were shown to be NP-hard [14], and a showcase for the field of *hardness of approximation*. There is theoretical evidence that the stable set problem is extremely hard to solve: A celebrated result of Håstad [10], derandomized by Zuckerman [24], shows that unless $P = NP$, there does not exist a $\left(n^{1-\varepsilon}\right)$-approximation for the stable set problem, where n is the number of vertices of G and $\varepsilon > 0$ is any fixed constant (here and throughout the paper, n denotes the number of vertices of the graph in analysis). From a

34th Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2014).
Editors: Venkatesh Raman and S. P. Suresh; pp. 187–198

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

parameterized perspective, the stable set problem is $W[1]$-hard (see e.g. [4]), *i.e.*, there is no polynomial time algorithm even if the cardinality of the optimal solution is considered as a fixed parameter.

When restricted to special graph classes the problem becomes more tractable. For instance, it is polynomial time solvable in perfect graphs and claw-free graphs [5, 8, 19], while it has a PTAS e.g. on planar graphs [2]. Foremost, it can be solved efficiently on bipartite graphs. In this case, the edge-node incidence matrix

$$A_G(e,u) = \begin{cases} 1 & \text{if } u \text{ is incident to } e, \\ 0 & \text{otherwise,} \end{cases}$$

is *totally unimodular*, see, e.g. [20]. By the Hoffman-Kruskal theorem [12], all the vertices of the natural linear programming relaxation for the stable set problem

$$\max \left\{ \sum_{v \in V} x_v \colon A_G\, x \leq \mathbf{1}, \quad x \geq \mathbf{0} \right\}$$

are then integral, and one can resort to algorithms for linear programming to efficiently solve the problem, also in its weighted variant. A 0/1-matrix is totally unimodular if the largest absolute value of any of its sub-determinants is at most 1. The guiding questions of our paper are:

> How well can the stable set problem be approximated if A_G is not totally unimodular? Is it possible to parametrize the approximability by the largest absolute value Δ of a sub-determinant of A_G?

Recall that $\Delta = 1$, i.e. the totally unimodular case, is equivalent to G having no odd cycles. Δ maintains a neat combinatorial interpretation also when it is greater than 1. Define the *odd cycle packing number* of G to be the cardinality of a maximum family of vertex-disjoint odd cycles of G. We denote it by $\operatorname{ocp}(G)$, omitting the dependence on G when it is clear from the context. Note that $\operatorname{ocp}(G) \leq n/3$ for all graphs G. As observed in [7], for a graph G with edge–node incidence matrix A_G, the largest absolute value of a sub-determinant is $\Delta = 2^{\operatorname{ocp}}$. Hence upper bounding Δ is tantamount to upper bounding the odd cycle packing number of G.

Our contribution. We show that the stable set problem can be very well approximated if $\operatorname{ocp}(G) = o(n/\log n)$. Beyond this, the approximation guarantee smoothly approaches the hardness result from [10] as ocp approaches $n/3$.

▶ **Theorem 1.** *The stable set problem on a graph G admits the following approximation algorithms:*

- *Polynomial time approximation scheme (PTAS) if $ocp(G) = o(n/\log n)$;*
- *$O\left(n^{1/p}\right)$-approximation if $ocp(G) \leq \frac{n}{2\delta p - 1}$ for p integer (possibly a function of n) and any constant $\delta > 1$.*

Surprisingly, these approximation guarantees can be obtained using simple greedy algorithms. Theorem 1 implies that stable set is fixed-parameter tractable (with the parameter being the size of the maximum stable set) when $\operatorname{ocp} = o(n/\log n)$, see Corollary 12, and that graphs where the hardness [10] is achieved can essentially be partitioned into triangles.

Before discussing each of the two algorithms of Theorem 1 separately, let us remark that both can be executed and will output a certificate of the approximation guarantee even

without the knowledge of the ocp of the input graph. The first algorithm runs in polynomial time on graphs with ocp $= o(n/\log n)$. However, it might run in super-polynomial time on graphs with ocp $= \Omega(n/\log n)$. The second algorithm always runs in polynomial time.

In order to make the first algorithm robust against malicious input, it would be sufficient to find the ocp of the input graph. Unfortunately, computing the odd cycle packing number of a graph is NP-hard and inapproximable up to $\log^{1/2-\varepsilon} n$ for all $\varepsilon > 0$ (see the discussion in [3]). The best-known approximation algorithm has only ratio $\sqrt{n}$ [15]. But, we can use the following weak separation theorem to distinguish between graphs with $\mathrm{ocp}(G) = \Omega(n/\log n)$ and graphs with $\mathrm{ocp}(G) = o(n/\log^2 n)$.

▶ **Theorem 2** (Weak separation). *Let $G(V, E)$ be a graph and $1 \leq c \leq n = |V|$ with c possibly depending on n. In time $O(n^4)$ we can conclude that $ocp(G) < \sqrt{cn}$, or that $ocp(G) \geq c$, and hence distinguish between graphs with $ocp(G) = \Omega(n/\log n)$ and graphs with $ocp(G) = o(n/\log^2 n)$.*

Thus, for most inputs to the algorithms from Theorem 1 we can efficiently check if they satisfy the condition on the ocp. Theorem 2 follows from a careful analysis of a simple greedy algorithm for ocp, see Theorem 3.

Considering hardness of approximation, we show that the stable set problem over G cannot be solved in polynomial time when ocp $= \Omega(\log^{1+\varepsilon} n)$ for all $\varepsilon > 0$, unless the exponential-time hypothesis is false (see Theorem 13). Hence, under the exponential-time hypothesis, Theorem 1 is best possible for graphs with ocp $\in \Omega(\log^{1+\varepsilon}(n)) \cap o(n/\log n)$ for each $\varepsilon > 0$.

The stable set problem, parameterized by the odd cycle packing number, can also be seen as a special case of integer programming parameterized by the absolute value of a sub-determinant of the constraint matrix. In particular, our hardness result provides an example of a non-polytime solvable (under the exponential-time hypothesis) integer program when the absolute value of the maximum sub-determinants is assumed to be of order $\Omega\left(n^{\log^{\varepsilon} n}\right)$.

In the weighted version of the stable set problem, a non-negative integer weight is associated to each vertex, and the goal is to find the set of non-adjacent vertices of maximum total weight. Extending our PTAS for the unweighted case we show that the weighted stable set problem on G admits a PTAS if ocp $= O(\sqrt{\log n/\log\log n})$ (see Theorem 15).

A key ingredient in all our algorithms is an idea of Györi *et al.* [9]. If a graph does not have small odd cycles, it is possible to find a small set of vertices that can be removed from the graph in order to make it bipartite. We extend the result of [9] to the vertex weighted case which supplies the necessary machinery for the PTAS for the weighted variant. This generalization is obtained by relating odd cycles of small *weight* with the *number* of vertices to be removed to make the graph bipartite. One easily checks that a variant relating odd cycles of small *weight* with the *weight* of vertices to be removed cannot exist.

Organization of the paper. We conclude this first section settling notation and highlighting related work. Section 2 is devoted to an improved approximation of ocp and Theorem 2. Section 3 is devoted to the proof of Theorem 1. We will first outline an easy algorithm (Algorithm 2) that provides the basic ideas and yields a first approximation result. This will prove the second part of Theorem 1. Then, we will discuss ideas on how to refine the analysis to prove the first part of Theorem 1. Section 4 deals with the weighted version of both Györi *et al.*'s theorem and the PTAS for the weighted stable set problem.

1.1 Notation

Throughout this paper we consider an undirected graph $G = (V(G), E(G))$. When G is clear from the context we will denote its vertex and edge sets by V and E respectively. For each $v \in V(G)$, let $N_i(v)$ denote the i-th neighborhood in a breadth-first search from v. For $S \subseteq V$, we denote by $G[S]$ the subgraph of G induced by S. For a weighted graph G and $v \in V(G)$ we use $w(v)$ to denote the weight of v and for a subset $S \subseteq V$ define $w(S) = \sum_{v \in S} w(v)$. All weights are assumed to be non-negative integers. Most of the other notation and definitions we employ are standard, and we refer the reader to textbooks for details on graphs, approximation algorithms, fixed-parameter tractability, and big-O notation (see for instance [21], [4], and [23]).

1.2 Related work

A dual concept to packing odd cycles is finding an *odd cycle transversal*, *i.e.*, the minimum number of vertices that have to be removed to bipartize the graph. We denote it by $\mathrm{oct}(G)$. Clearly for each graph G we have $\mathrm{ocp}(G) \leq \mathrm{oct}(G)$, and there are examples of graphs, called *Escher walls* [18], where $\mathrm{ocp} = 1$ and $\mathrm{oct} = \sqrt{n}$. In some special cases there are odd cycle transversals of small size. For instance, in planar graphs, it is known that $\mathrm{oct} \leq 6\,\mathrm{ocp}$ [16]. Györi *et al.* [9] showed that oct is also small when G does not have "short" odd cycles. More formally, for each $\varepsilon > 0$ (possibly a function of n), if G has no odd cycle of length at most εn, then $\mathrm{oct}(G) \leq f(\varepsilon)$ for some function f only depending on ε. This result and our new generalization are key components of our algorithms. Both computing ocp and oct is NP-hard, even when the input graph is planar [15, 6]. However, the two problems admit a $\sqrt{n}$ [15] and $\sqrt{\log n}$ [1] approximation, respectively.

The *exponential-time hypothesis* (ETH) [13] states that, for each $k \geq 3$, $k-$SAT cannot be decided with a sub-exponential time algorithm. An algorithm runs in *sub-exponential time* if it takes time $2^{o(n)}$ where n is the size of the instance. Impagliazzo *et al.*[13] also show that the ETH implies that there is no sub-exponential time algorithm for the stable set problem.

In the rest of the paper we will often use the fact that a shortest odd cycle in a graph can be found in time $O(n^3)$. This is standard, and can be found e.g. in [17].

2 Odd cycle packing

In this section, we show that a simple greedy algorithm provides a $\min\{\mathrm{ocp}, n/\mathrm{ocp}\}$-approximation for the ocp. Hence, when $\mathrm{ocp} = \Theta(\sqrt{n})$, it matches the $O(\sqrt{n})$-approximation by Kawarabayashi and Reed [15], and in all other cases improves over it.

Algorithm 1: Greedy algorithm for ocp

Input: a graph G.
Output: a family $\mathcal{C}$ of vertex-disjoint odd cycles of G.

1. Set $G' = G$, $\mathcal{C} = \emptyset$.
2. WHILE G' is not bipartite:
   i) Find a shortest odd cycle C of G'.
   ii) Set $\mathcal{C} = \mathcal{C} \cup \{C\}$ and $G' = G' \setminus C$.
3. Return $\mathcal{C}$.

▶ **Theorem 3.** *In time $O(n^4)$ Algorithm 1 finds a $\min\{ocp, \frac{n}{ocp}\}$-approximation for ocp.*

Proof. Let $\text{ocp} := \text{ocp}(G)$, $|V| = n$, $\mathcal{O}$ be the optimal solution, and $\mathcal{C} = \{C_1, \dots, C_t\}$ be the solution output by Greedy. As Algorithm 1 will find at least one odd cycle (if any), we know it is an ocp-approximation. So we only need to prove that $t\frac{n}{\text{ocp}} \geq \text{ocp}$ or, equivalently, $\text{ocp}^2 \leq tn$.

For $i = 1, \dots, t$, let $|C_i| = c_i$ and k_i be the number of cycles from $\mathcal{O}$ that intersect C_i but none of $C_1, \dots, C_{i-1}$. Note that $\sum_{i=1}^{t} k_i = \text{ocp}$. As all cycles from $\mathcal{O}$ contributing to k_i have length at least $|C_i| = c_i$, and as cycles from $\mathcal{O}$ are vertex-disjoint, we deduce

$$\sum_{i=1}^{t} k_i c_i \leq n. \tag{1}$$

As C_i can intersect at most $|C_i| = c_i$ vertex-disjoint odd cycles, we have

$$0 \leq k_i \leq c_i \qquad \text{for all } i. \tag{2}$$

We conclude that

$$\text{ocp}^2 = (\sum_{i=1}^{t} k_i)^2 \leq t \sum_{i=1}^{t} k_i^2 \leq t \sum_{i=1}^{t} k_i c_i \leq tn,$$

where the first inequality follows from Cauchy-Schwarz, and the others from (2) and (1) respectively. The complexity bound follows from the fact that we can find an odd cycle in a graph in time $O(|V|^3)$. ◀

In order to prove Theorem 2, it is sufficient to observe that, if the greedy algorithm outputs $\mathcal{C}$ of cardinality $t < c$, then we know $\text{ocp}(G) \leq \sqrt{tn} < \sqrt{cn}$. Otherwise, $\text{ocp}(G) \geq t \geq c$. The second part of the theorem follows by taking $c = n/\log^2 n$.

It is easy to see that Theorem 3 is essentially tight for each value of ocp. We give here the construction for $\text{ocp} = \sqrt{n}$: let G be a graph with odd cycles $C_0, C_1 \dots, C_{\sqrt{n}}$ all of length $\sqrt{n}$, such that C_0 intersects each of $C_1, \dots, C_{\sqrt{n}}$ in exactly one vertex, and $C_1 \dots, C_{\sqrt{n}}$ are pairwise vertex-disjoint. Then the greedy algorithm may pick C_0 only, while $\text{ocp}(G) = \sqrt{n}$. One easily generalizes this construction to other values of ocp.

Now let $\mathcal{C}$ be the family of all odd cycles of the graph, and let *f-ocp* be the optimum solution to the following natural LP relaxation of ocp:

$$\begin{aligned} \max \quad & \sum_{C \in \mathcal{C}} y_C & & \text{(ocp-LP)} \\ & \sum_{C : v \in C} y_C \leq 1 & & \text{for all } v \in V \\ & y_C \geq 0 & & \text{for all } C \in \mathcal{C}. \end{aligned}$$

With arguments very close to those used in the proof of Theorem 3 one can show the following, also improving over results from [15]. We defer details to the journal version of the paper.

▶ **Theorem 4.** *The greedy algorithm is a $\min\{ocp, \frac{|V|}{\text{f-ocp}}\}$ algorithm for the ocp and a $\min\{\text{f-ocp}, \frac{|V|}{\text{f-ocp}}\}$ for f-ocp. In particular, the integrality gap of the LP given by (ocp-LP) is bounded by $\min\{\text{f-ocp}, \frac{|V|}{\text{f-ocp}}\}$.*

3 Stable set

The main idea of all subsequent algorithms is the following. In a first step, delete *small* odd cycles. Due to the definition of small, this will only delete a small portion of the graph. If the remaining graph is bipartite, we can solve the stable set problem to optimality on this remainder. Otherwise, in a second step, we can leverage the fact that the remaining graph does not contain small odd cycles and obtain a large stable set.

The precise implementation of this second step depends on the size of the ocp. We will first show a general method that does not require to know the ocp of the input graph, *i.e.*, its runtime is independent of the ocp but the approximation guarantee depends on it (Algorithm 2). Then, assuming that ocp $= o(n/\log n)$ we will show how to improve this second step to obtain a PTAS (Algorithm 3 and 4).

3.1 A first approximation algorithm

Now, assume that G does not have an *odd* cycle of size at most $2k+1$. Lemma 5 states that G must have a large stable set (depending on k). We defer the proof to the journal version. A similar result, though with slightly different dependencies, was also obtained in [22].

▶ **Lemma 5.** *Let $k \in \mathbb{N}$ and G be a graph with no odd cycles of cardinality less or equal to $2k+1$ ($k \geq 1$). Then, there exists a stable set S of size $|S| > \frac{1}{3} n^{\frac{k}{k+1}}$ which can be found in time $O(n^{5/2})$.*

▶ **Corollary 6.** *Let G be a graph without odd cycles of cardinality less or equal to $2k+1$ ($k \geq 1$). Then Lemma 5 gives a $\left(3\ n^{1/(k+1)}\right)$-approximation algorithm to the stable set problem.*

Now, given any graph G, Algorithm 2 iteratively removes odd cycles of length up to some fixed value $2k+1$, and then applies Lemma 5 to obtain a large stable set. The optimal k will depend on the odd cycle packing number of the graph. Since we want to apply it to graphs whose ocp is not known a priori, we run the algorithm for all possible values of k. Let us remark that the approximation guarantee of Algorithm 2 is in terms of the ocp. However, we can compute an upper bound on the effective guarantee without knowing the ocp. This is because the bound on the approximation guarantee compares the obtained solution with the size of the entire graph.

Algorithm 2: Approximation algorithm for general ocp

Input: a graph G.
Output: a stable set of G.

1. For $k = 1, \ldots, \left\lfloor \frac{n-1}{2} \right\rfloor$ do:
   i) $G' = G$, $S_k = \emptyset$.
   ii) While there exists an odd cycle of cardinality at most $2k+1$ in G', delete it from G'.
   iii) If G' is empty, choose any vertex v in G and set $S_k = \{v\}$.
   iv) Else apply Lemma 5 to G' and obtain S_k.
2. Return the set S_k of maximum cardinality.

▶ **Theorem 7.** *Algorithm 2 has approximation guarantee*

$$\begin{cases} n & \text{if } ocp = \frac{n}{3} \\ 3n(n-(2p+1)ocp)^{\frac{1}{p+1}-1} & \text{if } ocp < \frac{n}{2p+1} \text{ for some } p \in \mathbb{N} \end{cases}$$

for the stable set problem and runs in time $O\left(n^5\right)$.

Proof. There are $\Theta(n)$ iterations. Finding a shortest odd cycle takes $O(n^3)$ time. Hence, the deletion step in each iteration takes at most $O(n^4)$ time. The application of Lemma 5 takes $O(n^{5/2})$ time. Overall, Algorithm 2 runs in time $O(n^5)$.

We now prove the approximation guarantee. If G' is empty at the end of step ii), the guarantee is clearly n. Otherwise, at most $(2k+1)$ocp vertices have been deleted. Hence, if $(2k+1)\text{ocp} < n$, then G' has at least $n-(2k+1)$ocp vertices and no odd cycle of cardinality at most $2k+1$. Therefore, the application of Lemma 5 yields a stable set of size at least $\frac{1}{3}\left(n-(2k+1)\text{ocp}\right)^{\frac{k}{k+1}}$. This implies an approximation guarantee of

$$n/\left(\frac{1}{3}\left(n-(2k+1)\text{ocp}\right)^{\frac{k}{k+1}}\right) = 3n\left(n-(2k+1)\text{ocp}\right)^{\frac{1}{k+1}-1},$$

under the condition that $(2k+1)\text{ocp} < n$. Given that $\text{ocp} < \frac{n}{2p+1}$ we obtain the claimed result with $k=p$. ◀

Note that this provides a smooth transition when ocp approaches $n/3$. This is formally stated in the next corollary and establishes the second part of Theorem 1 (the difference in the exponent is due to the offset of p in the bound on the ocp).

▶ **Corollary 8.** *Let G be a graph with $ocp(G) \leq \frac{n}{2\delta p+1}$ for a strictly positive integer p (possibly a function of n) and a constant $\delta > 1$. Then, Algorithm 2 yields a $O\left(n^{1/(p+1)}\right)$-approximation.*

Proof. Since $\delta > 1$ we have

$$\text{ocp} \leq \frac{n}{2\delta p+1} < \frac{n}{2p+1}.$$

Thus, we can use the result from Theorem 7 and obtain a guarantee of

$$3n(n-(2p+1)\text{ocp})^{-\frac{p}{p+1}} \leq 3n\left(n\left(1-\frac{2p+1}{2\delta p+1}\right)\right)^{-\frac{p}{p+1}} = 3n^{1/(p+1)}\left(\frac{2\delta p+1}{2p(\delta-1)}\right)^{\frac{p}{p+1}}.$$

Now, observe that $\frac{1}{2} \leq \frac{p}{p+1} \leq 1$ and $\left(\frac{2\delta p+1}{2p(\delta-1)}\right) \leq \frac{\left(\delta+\frac{1}{2p}\right)}{\delta-1} \leq \frac{\delta+1}{\delta-1}$. The claim immediately follows. ◀

Let us remark that Corollary 8 gives a $O(\sqrt{n})$-approximation for graphs that have $\text{ocp} \leq \frac{n}{3+\delta}$ for small constant $\delta > 0$. Note that in particular triangle-free graphs fall into this category and the best known approximation algorithm for these graphs has a guarantee of $O(\sqrt{n})$ [11]. Further, if $\text{ocp} = O\left(\frac{n}{\log n}\right)$ we obtain a constant factor approximation. We will see next that for smaller ocp we can even obtain an approximation scheme.

3.2 A PTAS for ocp $= o\left(n/\log n\right)$

We now restrict our attention to graphs G with $\text{ocp} = o\left(n/\log n\right)$, *i.e.*, we assume that there exists a function $f(n) \in o(1)$ such that the input graph G on n vertices has $\text{ocp}(G) \leq f(n)\frac{n}{\ln n}$. We give a PTAS for the stable set problem in these graphs. Again, the algorithm does not

require the knowledge of the ocp and hence can be run on any graph. However, the *existence* of such a function f is required to prove polynomial runtime.

The main new ingredient in this algorithm is to find a small odd cycle transversal X, *i.e.*, a small set of vertices whose removal bipartizes the graph. We use a result by Györi *et al.* [9] that gives a bound on $|X|$ that is, in a certain sense, independent of the size of the graph. Given the right parameter settings, we can ensure that $|X|$ is small and can therefore be removed from the graph. Solving stable set to optimality on the remaining bipartite graph then yields the solution.

We start by restating the theorem of Györi *et al.* [9]. The proof in [9] contains a minor, non-fatal error. We restate a correct version in Theorem 9 with adapted constants. The proof is constructive and can be turned into a polytime algorithm. We do not outline the proof of Theorem 9 here since it follows (albeit with new constants) also from the weighted version in Theorem 14.

▶ **Theorem 9** (from [9]). *Let $0 < \varepsilon \leq 1$ (possibly a function of n) and assume there exists no odd cycle of cardinality smaller than εn in G. Then, there is an odd cycle transversal X of G of size $|X| \leq \frac{48}{\varepsilon} \ln \frac{5}{\varepsilon}$ that can be found in $O(n^4)$ time.*

Note that the bound in Theorem 9 is only useful for $\varepsilon = \Omega(\log n/n)$. It will turn out later that this is the reason for the limitation of our PTAS to graphs with $\text{ocp} = o(n/\log n)$.

Algorithm 3 is a technical routine that will guarantee the existence of stable sets of a certain size, given an upper bound b on the ocp, and a parameter c that will depend on the approximation factor of the PTAS.

Algorithm 3: Subroutine for computing a large stable set

Input: graph G, parameters $b \geq c > 0$.
Output: a stable set of G.

1. Set $\varepsilon = \frac{c}{b}$, and let $G' = G$.
2. While there exists an odd cycle of cardinality smaller than $\varepsilon|V(G)|$ in G', remove it from G'.
3. If G' is not bipartite, find an odd cycle transversal X (Theorem 9) and remove it from G'.
4. Solve the stable set problem in G' and return the solution.

▶ **Lemma 10.** *Let G be a graph, and b and c chosen such that $b \geq \text{ocp}(G) > 0$ and $0 < c \leq b$. Then, Algorithm 3 applied to G, b, and c finds in time $O(n^4)$ a stable set of size at least*

$$\frac{1}{2}\left((1-c)n - \frac{48\,b}{c} \ln \frac{5\,b}{c}\right).$$

Proof. The algorithm removes at most ocp odd cycles of size at most εn in Step 2. Hence, G' has at least $n - \text{ocp}\ \varepsilon n \geq n - b\frac{c}{b}n = (1-c)n$ vertices. The claim now follows using the bound on $|X|$ from Theorem 9 and the fact that every bipartite graph admits a stable set on at least half of its vertices.

Finding a shortest odd cycle in G' takes $O(|V(G')|^3) \subseteq O(n^3)$ time. Since the removal of an odd cycle means the removal of at least three vertices, this procedure is only needed $O(n)$ times. Hence, Step 2 takes $O(n^4)$ time. Step 3 takes $O(n^4)$ time (Theorem 9). Solving maximum stable set in G' takes $O(n^{5/2})$ time [21, Cor. 19.3a]. In fact, the proof requires only one side of the bipartition which can be found in $O(n^2)$. Concluding, the runtime of Algorithm 3 is $O(n^4)$. ◀

Algorithm 4: PTAS for the unweighted case when $\operatorname{ocp} = o(n/\log n)$

Input: graph G and a parameter $\delta > 0$.
Output: a stable set of G.

1. Set $c = \frac{\delta}{3+2\delta}$ and $b = \frac{c^2 n}{50 \ln n}$.
2. Run Algorithm 3 with b and c and obtain a solution S. Let $\mathcal{C}$ denote the union of the cycles removed in Step 2 of Algorithm 3 and X denote the set found in Step 3 of Algorithm 3.
3. If $\frac{1}{2}|\mathcal{C}| + |X| \leq \delta |S|$, return S.
4. Otherwise, solve the maximum stable set problem by complete enumeration.

▶ **Theorem 11** (Polynomial time approximation scheme). *Let $1 \geq \delta > 0$ and G be a graph with $\operatorname{ocp}(G) = h(n)$, where $h(n) = o\left(\frac{n}{\log n}\right)$. Algorithm 4 is a $(1+\delta)$ approximation for the maximum stable set problem on G and runs in time $p(\delta) + O(n^4)$ for an appropriate function p depending only on δ.*

Proof. If Algorithm 4 returns a solution in Step 4, we obtain the optimal solution. Hence, assume that Algorithm 4 returns a solution in Step 3. We derive an upper bound on the size of the optimal solution $\operatorname{OPT}(G)$. Let G' denote the bipartite graph after the application of Step 3 of Algorithm 3. Recall that $G = G' \cup X \cup \mathcal{C}$. In each cycle of $\mathcal{C}$, the optimal solution can take at most half of the vertices. Note that S is the optimal solution on the bipartite graph G' (Step 4 of Algorithm 3). Hence,

$$\operatorname{OPT}(G) \leq \operatorname{OPT}(G') + |X| + \frac{1}{2}|\mathcal{C}| = |S| + |X| + \frac{1}{2}|\mathcal{C}| \leq (1+\delta)|S|.$$

Since $\operatorname{ocp}(G) = o(n/\log n)$ there exists a function $f \in o(1)$ with $\operatorname{ocp}(G) \leq f(n)\frac{n}{\ln n}$. Let N be the constant

$$N = \min\left\{ m \in \mathbb{N} : \frac{c^2 m}{50 \ln m} \geq 1 \text{ and } f(m') \leq \frac{c^2}{50} \text{ for all } m' \geq m \right\}.$$

We now prove that for $n \geq N$ the condition in Step 3 of Algorithm 4 will be satisfied. Hence, the algorithm only uses complete enumeration for constant size graphs, where the constant only depends on δ when we fix the function h in the statement of the theorem. Otherwise, it relies on Algorithm 3 which runs in time $O(n^4)$. This concludes the analysis of the running time.

Therefore, assume $n \geq N$. By the definition of N and b we have $b \geq 1$ and $b = \frac{c^2 n}{50 \ln n} \geq f(n)\frac{n}{\ln n} \geq \operatorname{ocp}(G)$. Thus, b is a valid upper bound for ocp. The size of each cycle in $\mathcal{C}$ is upper bounded by εn. Therefore $|\mathcal{C}| \leq \operatorname{ocp} \varepsilon n \leq b \varepsilon n = c n$. Furthermore, we have from Theorem 9 and the setting of b that

$$|X| \leq \frac{48b}{c} \ln\left(\frac{5b}{c}\right) \leq c n \frac{1}{\ln n} \ln\left(\frac{n}{\ln n}\right) \leq c n.$$

Plugging this into the bound of Lemma 10 gives $|S| = \operatorname{OPT}(G') \geq \frac{1}{2}\left((1-c)n - c n\right) = \frac{1}{2}(1-2c)\, n$. Finally, using $c = \frac{\delta}{3+2\delta}$ we obtain

$$\frac{|X| + \frac{1}{2}|\mathcal{C}|}{|S|} \leq \frac{cn + \frac{1}{2}cn}{\frac{1}{2}(1-2c)\, n} = \frac{3c}{1-2c} = \delta.$$

Thus, if $n \geq N$, Algorithm 4 returns a solution in Step 3. ◀

3.3 Fixed-parameter tractability

Algorithm 4 is in fact an *efficient* PTAS (see e.g. [4]), for its running time is at most $p(\delta) + O(n^4)$, for appropriate function p. Using standard arguments (see [4, Theorem 4.6]) we can use Algorithm 4 to conclude the following.

▶ **Corollary 12.** *Let $h : \mathbb{N} \to \mathbb{N}$ be a function such that $h(n) = o(n/\log n)$. Then, the stable set problem is FPT in the class of graphs with ocp $= h(n)$. More precisely, for a graph G in this class it can be solved in time $f(OPT) + n^4$, where OPT is the maximum stable set and f is an appropriate function depending only on OPT.*

3.4 Hardness

▶ **Theorem 13.** *If the ETH is true, then for each $\varepsilon > 0$, the maximum stable set problem cannot be solved in polynomial time for graphs with ocp $= \Omega((\log n)^{1+\varepsilon})$.*

A sketch of the proof is as follows: given a graph G, we can "blow it up" in order to obtain a new graph G' with much more vertices, but the same ocp. In particular, the ocp will be much smaller with respect to the number of vertices of the graph. Then, we show that solving the stable set problem for G' in polynomial time implies that we can solve the stable set problem for G in sub-exponential time, contradicting the ETH. We defer the details to the journal version.

4 Weighted stable set

4.1 Graphs without odd cycles of small weight are almost bipartite

In this section, we show that a graph without odd cycles of small weight has an odd cycle transversal of small cardinality. This generalizes Theorem 9 by Györi *et al.* [9]. The proof of Theorem 14 follows the same framework of the original: we iteratively find and delete "small" neighborhoods of vertices from a "small" cycle. However, the presence of weights implies some difficulties that are non-trivial to overcome. Let us mention two here: first, unlike in the unweighted case, not all vertices from a "small" cycle are a suitable starting point for this procedure. Second, it is not clear a priori how to define weighted neighborhoods appropriately. We postpone the full proof to the journal version. The weighted version also provides a proof of the unweighted case with different constants, but the same dependency on ε.

▶ **Theorem 14.** *Let G be a graph with n vertices and node weights w. Let $0 < \varepsilon \leq 1$ and assume there exists no odd cycle C of G with $w(C) < \varepsilon w(V)$. Then, there exists an odd cycle transversal X of G of size $|X| \leq \frac{96}{\varepsilon} \ln \frac{10}{\varepsilon}$ that can be found in $O(n^4 \log(w(V)))$ time.*

4.2 A PTAS for ocp $= O(\sqrt{\frac{\log n}{\log \log n}})$

In this section, we present a PTAS for the *weighted* stable set problem. Note that this time we need to know the ocp of the graph in advance. We only sketch the proof here and defer it to the journal version.

Similarly to the unweighted case, we iteratively remove odd cycles of small *weight* until we can find an odd cycle transversal X of small cardinality, whose existence is guaranteed by Theorem 14. There are two obstacles to overcome. First, the weight of the set X can be very high, so we cannot afford to simply remove it as in the unweighted case. Second, the

maximum weight of a stable set in the graph is not necessarily a constant fraction of the total weight of the graph. Thus we have to ensure that the total weight of the odd cycles that we remove is small with respect to the weight of the optimum solution.

We overcome the first obstacle by enumerating all possible stable sets on X. The second obstacle can be dealt with via the following observation. The input graph $G = (V, E)$ can be partitioned into ocp many odd cycles and one bipartite subgraph. By the pigeonhole principle, one of these subgraphs has weight at least $\frac{1}{\text{ocp}+1}w(V)$. Since the maximum stable set of an odd cycle C is at least $\frac{1}{3}w(C)$, we conclude that the maximum weight stable set of G has weight at least $\frac{1}{3\text{ocp}+3}w(V) \geq \frac{1}{6\text{ocp}}w(V)$. The combination of those two difficulties limits the applicability of our PTAS to graphs with $\text{ocp} = O(\sqrt{\log n / \log\log n})$.

▶ **Theorem 15.** *Algorithm 5 is a PTAS for the weighted stable set problem if* $ocp = O\left(\sqrt{\frac{\log n}{\log\log n}}\right)$.

Algorithm 5: PTAS for the weighted stable set problem in graphs with $ocp = O\left(\sqrt{\frac{\log n}{\log\log n}}\right)$

Input: graph G, vertex weights $w : V \to \mathbb{N}$, a parameter $\delta > 0$.
Output: a stable set of G of weight at least $\frac{1}{1+\delta}$ times the optimal value.

1. Set $\varepsilon = \frac{\delta}{6(1+\delta)\text{ocp}^2}$ for the remainder of the algorithm and use $G' = G$ as a copy.
2. While there exists an odd cycle of weight at most $\varepsilon w(G')$ in G', remove it from G'.
3. Apply Theorem 14 to find an odd cycle transversal X of G'.
4. Compute the maximum weight stable set in G':
   - For each stable set $\bar{S} \subseteq X$, compute the maximum weighted stable set S' in $G' \setminus (X \cup N(\bar{S}))$.
   - Return $\bar{S} \cup S'$ with maximum weight.

Acknowledgments. We thank Friedrich Eisenbrand for attracting our attention to the problem and for stimulating discussions. Yuri Faenza was supported by the German Research Foundation (DFG) within the Priority Programme 1307 Algorithm Engineering. Andres J. Ruiz-Vargas was supported by the Swiss National Science Foundation grants 200020-144531 and 200021-137574.

References

1 A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev. $O(\sqrt{\log n})$ Approximation Algorithms for Min UnCut, Min 2CNF Deletion, and Directed Cut Problems. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC'05, pages 573–581, New York, NY, USA, 2005.

2 B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, January 1994.

3 M. Di Summa, F. Eisenbrand, Y. Faenza, and C. Moldenhauer. On largest volume simplices and sub-determinants. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, California, USA, January 04-06, 2015*, 2015.

4 R. G. Downey and M. R. Fellows. *Parametrized Complexity*. Springer-Verlag New York, Inc., New York, NY, USA, 1999.

5 Y. Faenza, G. Oriolo, and G. Stauffer. Solving the weighted stable set problem in claw-free graphs via decomposition. *Journal of the ACM*, 61(4), 2014.

6 S. Fiorini, N. Hardy, B. Reed, and A. Vetta. Approximate min-max relations for odd cycles in planar graphs. *Mathematical Programming*, 110(1, Ser. B):71–91, 2007.

7 J. W. Grossman, D. M. Kulkarni, and I. E. Schochetman. On the minors of an incidence matrix and its smith normal form. *Linear Algebra and its Applications*, 218:213 – 224, 1995.

8 M. Grötschel, L. Lovász, and A. Schrijver. Stable sets in graphs. In *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*, pages 272–303. Springer Berlin Heidelberg, 1988.

9 E. Györi, A. V. Kostochka, and T. Luczak. Graphs without short odd cycles are nearly bipartite. *Discrete Mathematics*, 163(1–3):279–284, 1997.

10 J. Håstad. Clique is hard to approximate within $n^(1-\varepsilon)$. In *Acta Mathematica*, pages 627–636, 1996.

11 M. M. Halldórsson. Approximations of independent sets in graphs. In Klaus Jansen and José Rolim, editors, *Approximation Algorithms for Combinatiorial Optimization*, volume 1444 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg, 1998.

12 A. J. Hoffman and J. B. Kruskal. Integral boundary points of convex polyhedra. In *Linear inequalities and related systems*, Annals of Mathematics Studies, no. 38, pages 223–246. Princeton University Press, Princeton, N. J., 1956.

13 R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512 – 530, 2001.

14 R. M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

15 K.-I. Kawarabayashi and B. Reed. Odd cycle packing. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC'10, pages 695–704. ACM, 2010.

16 D. Král, J.-S. Sereni, and L. Stacho. Min-max relations for odd cycles in planar graphs. *SIAM Journal on Discrete Mathematics*, 26(3):884–895, 2012.

17 B. Monien. The complexity of embedding graphs into binary trees. In *Fundamentals of Computation Theory, FCT'85, Cottbus, GDR, September 9-13, 1985*, pages 300–309, 1985.

18 B. Reed. Mangoes and blueberries. *Combinatorica*, 19:267–296, 1999.

19 N. Sbihi. Algorithme de recherche d'un stable de cardinalite maximum dans un graphe sans etoile. *Discrete Mathematics*, 29(1):53 – 76, 1980.

20 A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.

21 A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume A. Springer, 2003.

22 J. B. Shearer. The independence number of dense graphs with large odd girth. *The Electronic Journal of Combinatorics*, 2, 1995.

23 D.B. West. *Introduction to Graph Theory*. Prentice Hall, 2000.

24 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*, STOC'06, pages 681–690, New York, NY, USA, 2006. ACM.