Parameterized Algorithms for Even Cycle Transversal

Pranabendu Misra[2], Venkatesh Raman[1], M.S. Ramanujan[1], and Saket Saurabh[1]

[1] The Institute of Mathematical Sciences, Chennai, India
{vraman,msramanujan,saket}@imsc.res.in
[2] Chennai Mathematical Institute
pranabendu@cmi.ac.in

Abstract. We consider a decision version of the problem of finding the minimum number of vertices whose deletion results in a graph without even cycles. While this problem is a natural analogue of the Odd Cycle Transversal problem (which asks for a subset of vertices to delete to make the resulting graph bipartite), surprisingly this problem is not well studied. We first observe that this problem is NP-complete and give a constant factor approximation algorithm. Then we address the problem in parameterized complexity framework with the solution size k as a parameter. We give an algorithm running in time $O^*(2^{O(k)})$ for the problem and give an $O(k^2)$ vertex kernel. (We write $O^*(f(k))$ for a time complexity of the form $O(f(k)n^{O(1)})$, where $f(k)$ grows exponentially with k.)

1 Introduction

Cycle hitting set problems like Feedback Vertex Set and Odd Cycle Transversal (OCT) are very well studied in graph theory, algorithms and complexity. In the Feedback Vertex Set problem and the Odd Cycle Transversal problem, we are given a graph $G = (V, E)$ and a positive integer k as inputs. The objective in these two problems is to check if there exists a subset $S \subseteq V$ of size at most k such that $G - S$ does not have a cycle (i.e. it is a forest) and does not have an odd cycle (it is bipartite), respectively. Both these problems are well studied in the realm of parameterized and approximation algorithms. In this paper, we consider a natural analogue of the Odd Cycle Transversal, namely the Even Cycle Transversal (EvenCT) problem and study this problem in the realm of parameterized complexity.

Even Cycle Transversal (EvenCT)

Instance: An undirected graph $G = (V, E)$ and a positive integer k
Parameter: k.
Problem: Does G have a set S of size at most k such that $G - S$ does not contain an even cycle?

M.C. Golumbic et al. (Eds.): WG 2012, LNCS 7551, pp. 172–183, 2012.

We start with some basic definitions in parameterized complexity. For a decision problem with an input of size n and a parameter k, the goal in parameterized complexity is to design an algorithm with a runtime $f(k)n^{O(1)}$ where f is a function of k alone. Problems which admit such algorithms are said to be *fixed parameter tractable* (FPT). A decision problem is said to have a *kernelization* algorithm, if there is a polynomial time algorithm that takes an instance (I, k) and returns an equivalent instance (I', k') such that, $k' \leq k$ and $|I'| \leq g(k)$, where g is a function of k alone. The theory of parameterized complexity was developed by Downey and Fellows [2]. For recent developments, see the book by Flum and Grohe [4].

The parameterized complexity of OCT was a long standing open problem, which was resolved in 2003 by Reed et al. [11], who developed an algorithm for the problem running in time $O^*(3^k)$. However, there has been no further improvement over this algorithm in the last 9 years though several reinterpretations of the algorithm have been published [7,9]. Only recently, an algorithm with a running time $O^*(2.32^k)$ has been obtained [10]. Thus, it is rather surprising that while OCT has attracted so much attention, EvenCT has so far been largely been ignored. Recently, Kakimura et al. [8] studied a generalization of EvenCT, called Subset Even Cycle Transversal. In this problem, apart from a graph $G = (V, E)$ and a positive integer k, we are also given a $T \subseteq V$ as the input. The objective is to determine whether there exists a vertex set $S \subseteq V$ of size at most k such that $G - S$ does not contain any even cycle whose intersection with T is non-empty. Observe that for $T = V$ this is precisely EvenCT. Their paper [8], shows that Subset Even Cycle Transversal is FPT, and thus EvenCT is also FPT. However this algorithm utilizes graph minor machinery, and the dependence of this algorithm on the parameter k is at least triply exponential. Hence, a second motivation is to find out if the special case EvenCT has a better FPT algorithm.

In this paper, we do a systematic study of EvenCT and obtain the following results.

- We show that EvenCT can be solved in time $2^{O(k)}n^{O(1)}$. To this end, we study the class of graphs $\mathcal{G}_e$, which is all graphs which do not contain even cycles as subgraphs. These graphs have been studied by Thomassen [13], who obtained a more general result that if a graph G does not contain cycles of length 0 modulo p for some fixed integer p then the treewidth of G is upper bounded by a function of p. This implies that the treewidth of a graph which does not contain even cycles is bounded by some constant. We show that, in fact $\mathcal{G}_e$ is even more structured. Any pair of two odd cycles in a graph from this class may intersect in at most one vertex. These kind of graphs are known as cactus graphs and their treewidth at most 2.
- We also show that EvenCT admits a quadratic kernel. That is, we obtain a polynomial time algorithm that, given an instance (G, k) of EvenCT returns an equivalent instance $(G' = (V', E'), k')$ such that $k' \leq k$ and $|V(G')| + |E(G')| = O(k^2)$. The kernelization algorithm is along similar lines as earlier

kernelization algorithms for Feedback Vertex Set [12] and the problem of Θ_c-Deletion [5].

This problem adds to the growing list of parity problems that are studied from the point of view of parameterized complexity. We also mention in passing that EvenCT is NP-complete and show that it has an approximation algorithm with factor 10.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph. A *cut vertex* is a vertex v such that $G - \{v\}$ contains two or more connected components than G does. A *block* in a graph is a maximal connected subgraph without a cut vertex. Thus blocks in a graph G are either an isolated vertex, an edge or a maximally-2-connected component. A *pendant block* is a block such that it has only one cut vertex. Removing a pendant block from a graph means, deleting all the vertices in that block except the cut vertex.

A *cactus graph* is a graph where each edge is part of at most one cycle. Equivalently a graph is a cactus graph if and only if each of its block is isomorphic to either K_1, K_2 or a cycle. An *odd cactus graph* is a cactus graph without any even cycles.

Lemma 1. [$*$][3] *If there are two cycles C_1 and C_2 in G such that C_1 and C_2 intersect in at least 2 vertices, then there is an even cycle in G.*

A set of vertices $S \subseteq V(G)$ is called an *ect* if $G - S$ does not contain an even cycle.

Lemma 2. *Let G be a graph and S be an ect of G. Then $(G - S)$ is an odd cactus graph.*

Proof. Lemma 1 implies that any pair of cycles in $(G - S)$ intersect in at most 1 vertex. Hence, $(G - S)$ is a cactus graph. Additionally, since $(G - S)$ excludes even cycles, it is an odd cactus graph.

3 NP-Completeness and Constant Factor Approximation

3.1 NP-Completeness

By a simple reduction from the NP-Complete Feedback Vertex Set problem, the NP-hardness of the problem is clear.

Theorem 1. [$*$] *The* EvenCT *problem is* NP*-Complete.*

[3] The proofs of results marked [*] will appear in the full version of the paper.

3.2 An Approximation Algorithm

We will show that EvenCT has a 10-approximation algorithm. First we prove the following easy lemma.

Lemma 3. [$*$] EvenCT *on cactus graphs is solvable in polynomial time.*

The Diamond Hitting Problem. The *diamond hitting problem* is defined as follows. Given a graph G, is there a subset S of at most k vertices of G such that, $(G - S)$ is a cactus graph. We need the following result by Fiorini et al.

Lemma 4 ([3], Theorem 6.6). *There is a factor* 9 *approximation algorithm for the diamond hitting problem.*

Using Lemma 3 and Lemma 4, we prove the following theorem.

Theorem 2. *There is a factor* 10 *approximation algorithm for* EvenCT*.*

Proof. Let G be the input graph. Let S be an optimum ect of G. Since $(G - S)$ is a cactus graph, S is also a solution to the diamond hitting problem on G. Hence, if T were an optimum solution to the diamond hitting problem on G, then $|T| \leq |S|$. We first apply Lemma 4 to obtain a factor 9 approximation to the diamond hitting problem on G. Let S_1 be the approximate solution obtained. Observe that $|S_1| \leq 9|S|$.

Since $H = G - S_1$, is a cactus graph, we can apply Lemma 3 to obtain an optimum ect S_2 for H in polynomial time. Since H is a subgraph of G, $|S_2| \leq |S|$. Therefore, the set $S_1 \cup S_2$ which is clearly an ect for G, has size at most $10|S|$.

This algorithm forms a critical part of the kernelization procedure in Section 5.

4 FPT Algorithm

In this section we give an algorithm for EvenCT which runs in time $O^*(2^{O(k)})$. Towards this, we first apply the technique of iterated compression. The idea is to start with a solution of size $k+1$ and try to 'compress' it to a solution of size k if possible.

We start with the compression version of the problem.

4.1 Compression Version

Compression EvenCT

Input:	Graph G, a positive integer k and an ect S of G such that $\|S\| = k+1$
Parameter:	k
Question:	Does G have an ect of size at most k ?

We fix an ect of G of size at most k (assuming one exists) and denote it by S^*. Let $Y = S \cap S^*$ and $N = S - Y$. We attempt to find an ect of the graph $(G - S) \cup N$ such that it has size at most $k - |Y|$ and is contained in $(G - S)$. Since the ect has to be disjoint from the set N, the graph N cannot contain even cycles and hence must be a cactus graph.

We give a branching algorithm to find the ect S^*. This algorithm works in two phases. In the first phase, we use an appropriate branching, at the end of which, every connected component C in $G - S$ is such that $C \cup N$ has no even cycles and all the edges from C to N are incident on the same connected component of N. In the second phase, we use the structure provided by the first phase to deal with the remaining even cycles in $(G - S) \cup N$.

R ecall that $G - S$ is an odd cactus graph. Let C be a connected component of $G - S$ such that either $C \cup N$ contains an even cycle, or there are at least 2 edges which have one end point in C, and the other end point in distinct components of N. In both cases, our aim is to find a 'minimal' path in C that satisfies the property (of forming an even cycle with N or having adjacency in more than two components of N). This will then suggest ways of branching (by picking vertices from this 'minimal' path).

Let P be any path in C and let r be any vertex of C. Let $v \in P$ be a vertex such that there is a path from r to v, which intersects P only at v, and let $R_P(r)$ be the set of all such vertices in P.

Lemma 5. *For any vertex r and any path P in C, $|R_P(r)| \leq 2$.*

Proof. Suppose that there are 3 vertices $\{u, v, w\}$ in $R_P(r)$. By definition of $R_P(r)$, there is a path from r to u which doesn't intersect $\{v, w\}$; similarly for v and w. Let T_1 be a minimal subtree of $(G-P) \cup R_P(r)$, that contains $\{u, v, w, r\}$, such that $\{u, v, w\}$ is the set of leaves of T_1. Let T_2 be a minimal sub-path of P which connects $\{u, v, w\}$. Then in $T_1 \cup T_2$, there are two cycles that have two or more common vertices. But then by Lemma 1, there is an even cycle in $(G - S)$, a contradiction.

Observation 1. *Given a vertex r and a path P in a component C of $G - S$, r is not reachable from any vertex in $P - R_P(r)$.*

Now, to identify a path with the required property, we define a partial order on all the paths in the component. Towards this end, we fix a root r_C for the component and measure distances of the paths from r_C. Let $dist(u, v)$ be the length of a shortest path between the two vertices u and v. Suppose for P $|R_P(r_C)| = 2$, then the *depth* of P, denoted by $d(P)$, is defined as the ordered pair $(dist(u, r_C), dist(v, r_C))$, where $R_P(r_C) = \{u, v\}$ and $dist(u, r_C) \geq dist(v, r_C)$. If $|R_P(r_C)| = 1$, then $d(P) = (dist(u, r_C), dist(u, r_C))$. As it will be clear from the context, we drop the symbol r_C from $R_P(r_C)$ hereafter. Given two paths P_1 and P_2, we say that P_1 is *smaller* than P_2 if either $d(P_1) > d(P_2)$ in the lexicographic order or, $d(P_1) = d(P_2)$ and P_1 is a sub-path of P_2. This relation induces a partial order on the set of all paths of a component C.

In the rest of the paper, by $P \cup N$ we refer to the graph obtained by considering the path P, the graph induced on N and all the edges between the vertices in P and the vertices in N. We call a path P a *branching path* if either $P \cup N$ contains an even cycle, or vertices in P have adjacencies to more than one connected component of N. Note that in the latter case, $P \cup N$ has fewer connected components than N has. By a *minimal branching path*, we mean a minimal element in the partial order defined above.
The following lemma shows that we can compute a branching path in polynomial time.

Lemma 6. [$*$] *Let H be a connected odd cactus graph such that $H \cup N$ either contains an even cycle or $H \cup N$ has fewer connected components than N has. Then we can find a path P of H in polynomial time such that P is a branching path.*

The next lemma shows that we can compute a minimal branching path in polynomial time. Furthermore, such a path Q has the crucial property that there is a subset of vertices A_Q in Q such that $|A_Q| \leq 6$ and there exists an optimum ect that either doesn't intersect Q or intersects only in a subset of A_Q. We call A_Q the set of important vertices in Q.

Lemma 7. [$*$] *Suppose P is a branching path in a connected component C of $G-S$. Then there is a branching path Q and a subset of its vertices A_Q containing at most 6 vertices such that, if any vertex v in $(Q - A_Q)$ is in some solution T, then there is a solution T' of size at most $|T|$ which doesn't contain v. Given P, we can find Q and A_Q in polynomial time.*

Using the above two lemmas we show the following,

Lemma 8. Compression EvenCT *is fixed-parameter tractable.*

Proof. We are given as input G and a solution S of size at most $k+1$. We wish to construct a new solution S^* of size at most k. We consider partitions of S into two sets Y and N, where $Y = S^* \cap S$ and $N = S - Y$.

For each such partition we delete Y from G. We then attempt to find an ect R of $(G-S) \cup N$ such that $R \subseteq V(G-S)$ and $|R| \leq k - |Y|$. For such a solution to exist, we require N to be an odd cactus graph. This can be easily checked by taking a block decomposition of N and checking for blocks not isomorphic to K_1, K_2 or an odd cycle.

To find an ect of $(G - S) \cup N$ we use a branching algorithm which we will analyze with the measure

$$\mu = (k - |Y| + \textit{number of components in } N.)$$

It is easy to see that $\mu \leq 2(k-|Y|)+1$. During the branching we update $(G-S)$ and N by deleting vertices or moving vertices from $(G - S)$ to N.

We first apply the following preprocessing rule. Let C be a connected component of $G - S$ such that there is at most one edge with one endpoint in C and

the other in N. Then, no vertex of C is part of an even cycle in $(G - S) \cup N$, and hence we delete C from $G - S$.

Let C be a connected component of $G - S$ such that $C \cup N$ has an even cycle or has fewer connected components than N has. We apply Lemma 6 to find a branching path Q. We then apply Lemma 7 on Q to find a minimal branching path P and A_P and branch on the vertices in A_P. Since $|A_P| \leq 6$, there are at most 7 branches, the first six branches each correspond to one of the vertices in A_p being added to the solution R and in the final branch, no vertex is chosen and we move the entire path P to N. Note that picking no vertex from P may not be feasible if $P \cup N$ contains an even cycle. When we add a vertex into the solution set R and delete it from the graph, μ decreases by 1. Also, since P was a branching path, moving P to N, reduces the number of connected components in N by at least 1. Hence, in every branch the measure μ drops by at least 1.

We do the above branching for every component in $(G - S)$. Eventually we'll reach a state where no component of $G - S$ satisfies the conditions of Lemma 6. Hence all the edges from each component are to the same connected component of N. Furthermore, if more than two edges from a component C are adjacent to a component of N, then it can be easily argued that there is an even cycle in $C \cup N$. Hence, every component of $G - S$ has edges to exactly one component of N and exactly two such edges. Let B_i be the set of vertices in a component C_i of $G - S$ such that they have a neighbour in N. Note that $|B_i| \leq 2$. We will show the following,

Claim. There exists an optimum solution to the compression EvenCT instance that, for any component C_i of $G - S$, is either disjoint from C_i or intersects C_i in a subset of B_i.

Proof. The claim follows from the observation that every even cycle intersecting C_i must also intersect b_1 and b_2, where b_1 and b_2 are the vertices in B_i.

For a pair of connected components C_i, C_j of $(G-S)$, let $H_{ij} = C_i \cup C_j \cup N$. We can check if H_{ij} contains an even cycle. If we find one, by the above claim we have a four-way branching on the vertices in $B_i \cup B_j$ and in each branch we add a vertex to R and delete it from the graph. In every branch μ drops by 1. We do this for every pair of connected components in $(G - S)$. Finally, let $(G - S) \cup N$ be the remaining graph. We will show that there are no even cycles left in this graph.

Claim. [$*$] If there is a cycle in $(G-S) \cup N$ passing through some $l > 2$ connected components of $(G - S)$, then there is a cycle passing through at most $l - 1$ components of $G - S$.

Claim. [$*$] If there is a cycle in $(G - S) \cup N$ passing through two connected components C_1, C_2 of $(G - S)$, then there is an even cycle in $C_1 \cup C_2 \cup N$.

The above two claims imply that $(G - S) \cup N$ now has no even cycles. For, if there was an even cycle, then it must pass through some l connected components of $G - S$. If $l = 1$, then there would be a connected component C of $G - S$ such

that $C \cup N$ contains an even cycle, a contradiction. Otherwise if $l \geq 3$, then we apply the first claim repeatedly to conclude that there are two components C_1 and C_2 such that $C_1 \cup C_2 \cup N$ contains a cycle which passes through both C_1 and C_2. Then by the second claim, there is an even cycle in $C_1 \cup C_2 \cup N$, a contradiction. Hence, we conclude that at the end of the above branching procedure we have constructed a solution R and the remaining graph is even cycle free.

For a partition of S into Y and N, the above branching takes time $O^*(7^{2(k-|Y|)})$. Summing over all the 2^{k+1} partitions of S we see that the overall algorithm takes time $\sum_{i=0}^{k} \binom{k+1}{i} O^*(7^{2k-2i}) = O^*(50^k)$.

Finally the following claim proves the correctness of the algorithm.

Claim. [$*$] Suppose S' were some solution to this EvenCT compression instance of size at most k. Let S^* be the solution constructed by the above algorithm. Then $|S^*| \leq k$.

4.2 FPT Algorithm for EvenCT

Theorem 3. EvenCT *parameterized by solution size has an FPT algorithm running in time* $O^*(50^k)$.

Proof. Suppose G is the input graph and k is the parameter. We arbitrarily order the vertices of G and define a sequence of graphs $G_{k+2}, G_{k+3}, \ldots, G_n$, where G_i is the induced subgraph on the first i vertices of G.

It is easy to see that if for some i, G_i doesn't have an ect of size at most k, then G doesn't have an ect of size at most k. And if G has an ect of size k, then each G_i has an ect of size at most k. We start with the graph G_{k+2} and let S_{k+2} be the first $k+1$ vertices. Clearly S_{k+2} is an ect of size $k+1$ for G_{k+2}. We use Lemma 8 to compute a solution S of size at most k for G_{k+2}. If we fail, then there is no ect of size k for G and so we answer NO. And if we succeed then $S_{k+3} = S \cup \{v_{k+3}\}$ is an ect of size $k+1$ for G_{k+3}. We then repeat the above process.

Note that there are at most $O(n)$ iterations and each iteration takes time $O^*(50^k)$. So this algorithm runs in time $O^*(50^k)$.

5 Polynomial Kernel

In this section we give a quadratic kernel for EvenCT. We first introduce reduction rules which allow us to establish a bound on the maximum degree of the graph. During this process we may introduce parallel edges in the graph, which we can remove via a simple preprocessing. Following that we use the bounded degree of the graph to design further reduction rules, to obtain the kernel.

5.1 Bounding the Degree of the Graph

Definition 1. *For a vertex v, a set of even cycles is called a* flower passing through v *if the even cycles intersect only at v. If the number of even cycles in the set is t, then this set is called a* t-flower *passing through v. Each even cycle of a flower is called a* petal.

We will now use an approach similar to that in [5] and [12] to design reduction rules to eliminate vertices of high degree. We begin by proving a few lemmas which will be used crucially in designing a reduction rule that removes high degree vertices.

Lemma 9. [$*$] *Given a graph G such that $tw(G) = r$ for some constant r, and a vertex v in G, we can find in linear time, the size of the largest flower passing through v in G.*

Lemma 10. [$*$] *Given a graph $G = (V, E)$ let $v \in V$ be a vertex such that $G - \{v\}$ is an odd cactus graph, and let k be an integer. Then, in polynomial time, we can either compute an ect X of size $O(k)$ in G such that X is disjoint from v or conclude that there is a $k+1$-flower passing through v.*

Lemma 11. [$*$] *There is a polynomial time procedure, which, for every vertex v in the graph, either returns a set S_v such that $v \notin S_v$, $|S_v| = O(k)$ and S_v is a solution or conclude that there is a $k+1$-flower passing through v.*

We now move ahead to the description of the reduction rules.

Reduction 1. *If there are more than 2 parallel edges between a pair of vertices, delete all but 2 of them.*

Reduction 2. *For every vertex v, apply Lemma 11. If some vertex has a $(k+1)$-flower passing through it, then delete this vertex from the graph and reduce k by 1.*

Correctness. The correctness follows from the fact that such a vertex must be part of every solution of size at most k.

Definition 2. *For a vertex v, let $C_1^v, \ldots, C_r^v$ be the set of connected components of $G - (S_v \cup \{v\})$ which are adjacent to v in G.*

Observation 2. *A vertex v cannot have more than 2 neighbors in any component C_i^v.*

Proof. The proof follows from Lemma 1.

Reduction 3. *If there is a component C_i^v which does not have an edge to S_v, then we delete the vertices in this component.*

Correctness. The correctness follows from the fact the vertices of this component cannot be part of an even cycle in the given graph.

Definition 3. *Define a graph $G_v = (A, B, E)$ to be the bipartite graph where the vertices in A correspond to the components $C_1^v, \ldots, C_r^v$, and the vertices in B correspond to the vertices in S_v. We add an edge between two vertices $a_i \in A$ and $u \in B$ if there is an edge from the component C_i^r to the vertex u.*

We now state without proof (a slightly weaker form of) the well known q-expansion lemma which we will use to define the high degree vertices. This lemma is a generalization of a lemma in [12]. A q-star is defined as a star with q leaves. We say that a vertex v has a q-star in the vertex set T if v is the center of a q-star and the leaves of the star all lie in T.

Lemma 12. *Let q be a positive integer and let G be a bipartite graph with vertex bipartition $A \uplus B$. If there are no isolated vertices in B and $|B| > q|A|$, then there are non-empty vertex sets $S \subseteq A$ and $T \subseteq B$ such that S has $|S|$ many vertex disjoint q-stars in T and S separates the vertices of T from the rest of the graph. Furthermore, the sets S and T can be computed in polynomial time.*

Reduction 4. *If $deg(v) \geq 8|S_v|$, then compute a 3-expansion set $X \subseteq B$ in the graph $G_v = (A, B, E)$. Delete the components corresponding to the neighbors of X, and add parallel edges between v and u for every vertex $u \in X$.*

Note that, in each application of this rule, the degree of v decreases by at least $|X| > 0$. When the Rule 4 cannot be applied on any vertex in the graph, then the maximum degree of the graph is bounded by $O(k)$. Further note that because of Rule 1, there can be at most 2 parallel edges between any two vertices u and v. For the sake of convenience, we would like to deal with simple graphs and hence we do the following. For each pair of vertices u and v in G with 2 parallel edges, replace one of the edges by a path (u, a, b, v) of length 3 where a and b are two new vertices. It is easy to see that this process preserves the bound on the degree and the solution size, while resulting in a simple graph.

5.2 Bounding an Irreducible Yes Instance

Now, we prove a bound on the size of an irreducible Yes instance. At this point, we have a graph G where every vertex has $O(k)$ neighbors, and an approximate solution A of size $O(k)$. Note that the number of edges going across the cut $(G - A, A)$ is bounded by $O(k^2)$. Since $G - A$ is an odd cactus graph, we can find a block decomposition of this graph [6] and the corresponding *block graph* T [1]. T is a forest which contains a vertex for every block and every cut-vertex in $G - A$. There is an edge between two vertices u and v, where u corresponds to a cut-vertex and v corresponds to a block B_v in $G - A$, if $u \in B_v$ in the graph $G - A$. We call a vertex of $G - A$ an *affected vertex* if it is adjacent to a vertex in A. We call a block of $G - A$ an *affected block* if it contains a cut vertex.

Reduction 5. *Delete any pendant block in $G - A$ which is not affected.*

Correctness. It follows from the observation that such blocks are never part of any even cycle in G.

Reduction 6. *If there is a maximal path P in $G - A$ such that every vertex is of degree two and is unaffected, except for the endpoints which could either be affected vertices or be vertices of degree three or more, then replace P by a path of length two (or one) if this path is of even (or odd) length.*

Correctness. It follows from the observation that we can substitute for any such vertex by one of the endpoints, and the replacement doesn't change the parity of any cycle in G.

Definition 4. *Consider a path P of degree two vertices in T such that no vertex in P corresponds to an affected vertex or an affected block in $G - A$. We call the corresponding sequence of blocks $B_1, \ldots B_l$ a* chain *in $G - A$.*

It has the property that, no vertex in $\cup_{i=1}^{l} B_i$ has a neighbour in A, and every block has exactly two cut vertices. We call the two cut vertices at the two ends of a chain, the endpoints of the chain.

Reduction 7. *If there is a chain such that it consists of a single block which is an odd cycle of size greater than three, or contains at least two blocks with one of them an odd cycle, then replace this chain with a triangle which is formed by the two endpoints of the chain and a new vertex.*
If the chain contains no odd cycles, then the chain is a path in $G - A$. Furthermore, suppose that this chain contains at least three blocks. Then replace this path by a path of length two(or one) if this path is of even (or odd) length.

Correctness. The correctness follows from the fact that any vertex of the chain which is part of the solution can be simply replaced by one of the endpoints of the chain. □

Lemma 13. *The number of affected vertices in $G - A$ is $O(k^2)$ and the number of affected blocks is $O(k^2)$.*

Proof. The bound on the number of affected vertices follows from the fact that $|A| = O(k)$ and every vertex in A has a degree $O(k)$. This clearly implies the second statement.

The following two lemmas give a bound on the number of vertices in the block graph T, and then on the number of vertices in $G - A$.

Lemma 14. [$*$] *The number of vertices in the block graph is $O(k^2)$.*

Lemma 15. [$*$] *There are $O(k^2)$ vertices in $G - A$.*

Using the above lemma and the fact that $|A| = O(k)$ we have the following theorem.

Theorem 4. EvenCT *parameterized by the solution size has an $O(k^2)$ vertex kernel.*

References

1. Diestel, R.: Graph Theory, Graduate Texts in Mathematics, 3rd edn., vol. 173. Springer, Heidelberg (2005), http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/GraphTheoryIII.pdf
2. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, New York (1999)
3. Fiorini, S., Joret, G., Pietropaoli, U.: Hitting Diamonds and Growing Cacti. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 191–204. Springer, Heidelberg (2010)
4. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006)
5. Fomin, F.V., Lokshtanov, D., Misra, N., Philip, G., Saurabh, S.: Hitting forbidden minors: Approximation and kernelization. In: STACS, pp. 189–200 (2011)
6. Hopcroft, J., Tarjan, R.: Algorithm 447: efficient algorithms for graph manipulation. Commun. ACM 16, 372–378 (1973), http://doi.acm.org/10.1145/362248.362272
7. Hüffner, F.: Algorithm engineering for optimal graph bipartization. J. Graph Algorithms Appl. 13(2), 77–98 (2009)
8. Kakimura, N., Ichi Kawarabayashi, K., Kobayashi, Y.: Erdös-pósa property and its algorithmic applications: parity constraints, subset feedback set, and subset packing. In: SODA, pp. 1726–1736 (2012)
9. Lokshtanov, D., Saurabh, S., Sikdar, S.: Simpler Parameterized Algorithm for OCT. In: Fiala, J., Kratochvíl, J., Miller, M. (eds.) IWOCA 2009. LNCS, vol. 5874, pp. 380–384. Springer, Heidelberg (2009)
10. Loksthanov, D., Narayanaswamy, N.S., Raman, V., Ramanujan, M.S., Saurabh, S.: Faster Parameterized Algorithms using Linear Programming. ArXiv e-prints (March 2012)
11. Reed, B.A., Smith, K., Vetta, A.: Finding odd cycle transversals. Oper. Res. Lett. 32(4), 299–301 (2004)
12. Thomassé, S.: A quadratic kernel for feedback vertex set. In: SODA, pp. 115–119 (2009)
13. Thomassen, C.: On the presence of disjoint subgraphs of a specified type. Journal of Graph Theory 12(1), 101–111 (1988)