On the Algorithmic Effectiveness of Digraph Decompositions and Complexity Measures

Michael Lampis[1], Georgia Kaouri[2], and Valia Mitsou[1]

[1] City University of New York
[2] National Technical University of Athens
{mlampis,vmitsou}@gc.cuny.edu, gkaouri@corelab.ntua.gr

Abstract. We place our focus on the gap between treewidth's success in producing fixed-parameter polynomial algorithms for hard graph problems, and specifically Hamiltonian Circuit and Max Cut, and the failure of its directed variants (directed tree-width [9], DAG-width [11] and kelly-width [8]) to replicate it in the realm of digraphs. We answer the question of why this gap exists by giving two hardness results: we show that Directed Hamiltonian Circuit is $W[2]$-hard when the parameter is the width of the input graph, for any of these widths, and that Max Di Cut remains NP-hard even when restricted to DAGs, which have the minimum possible width under all these definitions. Our results also apply to directed pathwidth. (Eligible for best student paper)

Keywords. Treewidth, Digraph decompositions, Parameterized Complexity.

1 Introduction

Treewidth, first introduced by Robertson and Seymour in [13], has been one of the most successful tools in the research for efficient algorithms for hard graph problems in the last 15 years. Intuitively, treewidth allows us to distinguish graphs that have a relatively simple (tree-like) structure, and exploit that structure to solve a plethora of otherwise intractable problems, usually by employing a dynamic programming technique. For an introduction to the notion of treewidth see Bodlaender's excellent survey papers [4, 3, 2].

One of the most celebrated theorems in the area of treewidth is Courcelle's theorem which states that every graph property that can be expressed in monadic second order logic can be decided in linear time on graphs of bounded treewidth [5]. Beginning from this starting point, algorithms for many hard graph problems have been devised using treewidth. They almost invariably have running times of the form $O(f(k) \cdot n)$, where k is the treewidth of the input graph and f some exponential or super-exponential function which represents the complexity of solving the problem exhaustively on k vertices. Thus, not only is the running time polynomial for fixed k, but also that the combinatorial explosion is confined to k. This has led treewidth to become one of the cornerstones of parameterized complexity theory, a theory which describes the distinction between algorithms with running times of the form $O(f(k) \cdot n^c)$, where c is a constant (called fixed-parameter tractable or FPT) and algorithms of the form $O(n^{g(k)})$. For an introduction to parameterized complexity see the monograph by Downey and Fellows [6] or the introductory books by Niedermeier [10] and by Flum and Grohe [7].

Several attempts have been made recently to generalize the notion of treewidth to directed graphs. The motivation behind this line of research is that, although it is possible to solve many hard problems on digraphs when the underlying undirected graph has low treewidth by using traditional tree decompositions, this approach sacrifices a great deal of generality. A problem which demonstrates this to a great degree is DIRECTED HAMILTONIAN CIRCUIT. This problem is trivial when the input graph is a DAG, but there exist DAGs of unbounded treewidth if the direction of the edges is ignored. Thus, it is desirable to come up with an alternative measure of digraph complexity which better characterizes the class of digraphs where hard problems become tractable. It should be noted at this point that HAMILTONIAN CIRCUIT admits an FPT solution with a treewidth based algorithm, therefore a logical target when defining a digraph complexity measure would be to achieve fixed-parameter tractability for DIRECTED HAMILTONIAN CIRCUIT as well.

Previous work. The most notable variations of treewidth for digraphs that have been proposed in the past are probably directed treewidth [9], DAG-width [11] and kelly-width [8]. All these three measures can be viewed as good generalizations of treewidth in the sense that, if we take an undirected graph and replace each edge with two opposite directed edges the width of the new digraph will be the same for all three definitions and equal to the treewidth of the original graph. Directed treewidth is the most general of the three, in the sense that a graph of bounded kelly-width or DAG-width will also have bounded directed treewidth, while the converse may not be true. Also DAG-width and kelly-width are conjectured to be only a constant factor apart on any graph ([8]).

The most important positive result of directed treewidth (which can be extended to all the three measures) is an algorithm that solves DIRECTED HAMILTONIAN CIRCUIT in $O(n^k)$ time, k being the width of the input graph. Nevertheless, this algorithm is still far from the performance of the best treewidth-based algorithm for HAMILTONIAN CIRCUIT, which runs in fixed-parameter linear time. Unfortunately, the reason for this distance is not addressed in [9] or in [8] where another algorithm (of the same complexity) for this problem is given. In addition, the few already known algorithmic results on these measures don't seem to indicate that they are likely to achieve a level of success comparable to treewidth, as no FPT algorithms are known for any hard digraph problems. Of course, it could be conceivable that this is due to a lack of effort so far, since digraph decompositions have been introduced much more recently than treewidth.

A related measure is directed pathwidth. Just as pathwidth is a restriction of treewidth in the undirected case directed pathwidth is a restriction of all the previously mentioned directed measures, thus having even greater algorithmic potential. However, to the best of our knowledge no such results have been shown for directed pathwidth. In [1] it is shown that a cops-and-robber game is equivalent to directed path-width and that there always exists an (almost) optimal monotone strategy. It is worthy of note that, unlike the undirected case where treewidth and pathwidth are generalizations of different graph topologies (trees and paths respectively) in the directed case all the measures we have mentioned are based on the concept of DAGs as the simplest case.

Our contribution. In this paper we try to address the question of whether the already proposed digraph complexity measures will be able to match the success of treewidth. Our answer is given in the form of two negative results, which show that the lack of FPT algorithms for Directed Hamiltonian Circuit and Max Di Cut is not due to a lack of effort, but because such algorithms can not exist (under some widely believed complexity assumptions).

Our first result concerns Directed Hamiltonian Circuit which we show to be $W[2]$-hard when the parameter is the width of the input graph for any of the mentioned widths. Under the assumption that $W[2] \neq \text{FPT}$ this implies that no FPT algorithm is possible. Therefore, under this standard complexity assumption, our result implies that no significant improvement is possible for the $O(n^k)$ algorithms of [9] and [8].

Our second result concerns Max Di Cut, for which we show APX-hardness even when we restrict the problem to DAGs and all edges have uniform weights. This is a result that is interesting in its own right, and it is rather surprising that it was not known until now, as Max Di Cut is a widely studied problem. It is also very relevant in our case for two reasons: First, DAGs have the lowest possible width for all the widths we have mentioned, therefore our proof implies that none of them can help with Max Di Cut. Second, using (undirected) treewidth leads to efficient FPT algorithms for both Max Cut and Max Di Cut. Thus, this result helps draw further contrast between the performance of treewidth and its directed variants.

Although our results are negative, they succeed in illuminating some fundamental weaknesses in the already proposed digraph measures, and thus they show the way to a possible future digraph measure that might be able to overcome them. Therefore, we believe that they serve as a starting point in a renewed search for a successful digraph complexity measure that might yet manage to at least partially match treewidth's success.

The rest of this paper is structured as follows: In Section 2 we give some necessary definitions and preliminary notions. In Section 3 we demonstrate the hardness result for Directed Hamiltonian Circuit. In Section 4 we prove the hardness of Max Di Cut. Finally, in Section 5 we conclude with some discussion and directions to further research.

2 Definitions and Preliminaries

First, let us give the definitions of the two problems that will be our focus.

Definition 1. *The* Directed Hamiltonian Circuit *problem is that of deciding whether there exists a permutation* $(v_1, v_2, \dots, v_n)$ *of the vertices of an input digraph* $G(V, E)$ *s.t.* $\forall i \in \{1, \dots, n-1\}$ $(v_i, v_{i+1}) \in E$ *and* $(v_n, v_1) \in E$.

Definition 2. *The* Max Di Cut *problem is the following: given a digraph* $G(V, E)$ *and a weight function on the edges* $w : E \to \mathbb{N}$, *find a partition of* V *into two sets* V_0 *and* V_1 *so that the weight of the edge set* $C = \{(u, v) \mid u \in V_0, v \in V_1\}$ *is maximized. That is, the objective is to maximize* $\sum_{e \in C} w(e)$.

Max Di Cut was shown APX-hard in [12]. In Section 4 we show APX-hardness for the problem's restriction to DAGs. Then we show that APX-hardness also holds for the cardinality version of the problem restricted to DAGs.

We should also give the definitions of the two problems that will be the starting points of our reductions.

Definition 3. Dominating Set *is the problem of finding a minimum cardinality subset of vertices* D *of an undirected graph* $G(V, E)$ *s.t. any vertex in* $V \setminus D$ *has a neighbor in* D.

When a vertex $u \in D$ is a neighbor of a vertex v, we will say that u dominates v. We will also follow the convention of saying that any vertex in D dominates itself. We will make use of the well-known result that Dominating Set is $W[2]$-complete when the parameter k is the size of the dominating set we are looking for ([6]).

Definition 4. NAE3SAT *is the problem of finding a truth assignment which, for every clause of an input 3CNF formula, assigns the value true to at least one literal, and the value false to at least one literal.*

We follow the convention of saying that a clause is satisfied in the NAESAT sense, or simply satisfied, when a truth assignment assigns different truth values to two of its literals. We will mainly be concerned with the maximization version of NAE3SAT where the objective is to find a truth assignment that satisfies as many clauses as possible. This variant was shown to be APX-hard in [12].

We have already mentioned that directed pathwidth can be defined in terms of a cops-and-robber game. The game's definition is the following:

Definition 5. *The* k*-cop invisible-eager robber game is the game where* k *cops attempt to catch an invisible robber hiding in a vertex of a digraph* G. *The cops are stationed on vertices of* G *and a cop can move by removing himself from the graph and then "landing" on any other vertex. The robber can move at any time and he is allowed to follow any directed path of* G, *under the condition that he does not enter vertices occupied by stationary cops.*

We say that k cops have a monotone strategy to win this game when they have a strategy such that the robber can never visit a vertex previously occupied by a cop. In [1] it was shown that k cops have a monotone strategy on a graph G iff the graph has directed pathwidth k.

Kelly-width, DAG-width and directed treewidth have also been shown to be connected to similar games, restricted to monotone strategies. In fact, DAG-width is equivalent to the above game but with the robber being visible, while kelly-width is equivalent to the above game but with the robber only being allowed to move when a cop enters his vertex. Using the approximate connection between directed treewidth and a similar game it was shown in [8] that the directed treewidth of a graph is upper-bounded by its kelly-width multiplied by a constant.

It is not hard to infer from these results that, since the robber is stronger in the game related to directed pathwidth, a graph G will have higher pathwidth than any of the other widths. Since we are interested in proving hardness results, it will therefore suffice to show that a problem is hard for graphs of small directed pathwidth and hardness for the other widths will directly follow.

3 Directed Hamiltonian Circuit

In this section we focus on the Directed Hamiltonian Circuit problem, a problem which can be solved using directed treewidth in $O(n^k)$ time ([9]). Of course this algorithm also applies to DAG-width, kelly-width and directed pathwidth, as they are restrictions of directed treewidth. In addition, another $O(n^k)$ algorithm for this problem tailored for kelly-width is given in [8]. Thus, a significant gap exists between the performance of treewidth, which is fixed-parameter polynomial on the corresponding undirected problem and the performance of its directed variants. We show that this is a gap that can not be bridged unless $W[2] = FPT$, by demonstrating that Directed Hamiltonian Circuit is $W[2]$-hard when the parameter is any of these widths.

The hardness proof for Directed Hamiltonian Circuit will be a parameterized reduction from the naturally parameterized version of Dominating Set.

Theorem 1. *The parameterized versions of* Directed Hamiltonian Circuit*, where the parameter is the directed treewidth, kelly-width, DAG-width or directed pathwidth of the input graph, are $W[2]-hard$.*

Proof. We will show a parameterized reduction from the naturally parameterized version of Dominating Set, where the parameter k is the size of the set by constructing a digraph whose directed pathwidth is bounded by a function of k s.t. the digraph will be Hamiltonian iff the original graph had a dominating set of size k.

Suppose we are given a graph $G(V,E)$ with $V = \{1,2,\ldots,n\}$.

Our digraph G' has vertex set $V' = V_1 \cup V_2 \cup V_3$ where

1. $V_1 = \{u_1, u_2, \ldots, u_k\}$.
2. $V_2 = \{v_1, v_2, \ldots, v_n\}$.
3. $V_3 = \{g_{(i,j,l)} \mid i \in \{1,\ldots,n\}, j \in \{0,\ldots,d(i)\}, l \in \{In, Out\}\}$. Here $d(i)$ denotes the degree of vertex i in G.

$E(G')$ consists of the following sets of directed edges

1. $E_1 = \{(u_i, v_j) \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, n\}\}$.
2. $E_2 = \{(v_i, v_{i+1}) \mid i \in \{1, \ldots, n\}\}$ (where we consider $n + 1$ to be the same as 1).
3. $E_3 = \{(g_{(i,j,In)}, g_{(i,j+1,Out)}) \mid i \in \{1, \ldots, n\}, j \in \{0, \ldots, d(i)\}\}$, (where we consider $d(i) + 1$ to be the same as 0).
4. $E_4 = \{(g_{(i,j,Out)}, g_{(i,j,In)}) \mid i \in \{1, \ldots, n\}, j \in \{0, \ldots, d(i)\}\}$.
5. Finally, E_5 contains the following edges: For any vertex a of the original graph, let $j_0, j_1, \ldots, j_{d(a)}$ be the vertices of G a dominates in lexicographic order. Also, let $p(a, i)$ denote the number of vertices that dominate a vertex i and come before a in lexicographic order. Then the edge $(v_a, g_{(j_0,p(a,j_0),In)})$ and the edges $(g_{(j_i,p(a,j_i),Out)}, g_{(j_{i+1},p(a,j_{i+1}),In)})$ for all $i < d(a)$ are included in E_5. Finally, the edges $(g_{(j_{d(a)},p(a,j_{d(a)}),Out)}, u_i)$ for all $i \in \{1, \ldots, k\}$ are also included in E_5.

Let us now discuss the basic idea behind this construction, before we get into more details. Our digraph G' consists of three parts: a constraint part V_1, a choice part V_2 and a satisfaction part V_3. V_1 functions as a constraint part because it only has k vertices and the only edges going into V_2 originate here, thus forcing us to enter the choice part exactly k times. A Hamiltonian tour will leave V_2 k times. The vertices from which it leaves V_2 must be (as we will prove) a dominating set of G, and that is why V_2 is the choice part. Finally, V_3 is arranged in such a way that it can only be traversed in a Hamiltonian way if the choice made in V_2 is indeed a dominating set.

We will call the group of vertices $g_{(i,j,l)}$ for a specific i, the gadget C_i. The crucial part of this reduction is the way the gadget C_i works. Notice that the gadget's vertices induce a directed cycle. Also, the only way to enter this cycle is through an *In* vertex, and the only way to leave is through an *Out* vertex. Suppose that a Hamiltonian tour enters a gadget C_i m times and that $X \subseteq \{0, \ldots, d(i)\}$ is the index set of the *In* vertices that were used. Then it must also be the index set of the *Out* vertices used. To see that, suppose that $X = \{j_1, j_2, \ldots, j_m\}$ in lexicographic order. When entering from $g_{(i,j_1,In)}$ the tour has no choice but to proceed to $g_{(i,j_1+1,Out)}$. Then if $j_2 \neq j_1 + 1$ the tour must move to $g_{(i,j_1+1,In)}$, because if it were to exit this vertex would be impossible to visit in the future. Using this argument again can exclude the possibility of this part of the tour exiting through any vertex other than $g_{(i,j_2,Out)}$. Similarly, the path that starts at $g_{(i,j_2,In)}$ will exit at $g_{(i,j_3,Out)}$ and so on, with $g_{(i,j_m,In)}$ exiting through $g_{(i,j_1,Out)}$. This procedure covers all the vertices of the gadget, therefore we proved that for any set of entry vertices X the gadget can be traversed in a way that does not exclude the existence of a Hamiltonian tour of the whole graph iff X corresponds also to the exit vertices used.

Let us now make use of the property we just established. Suppose that G does not have a dominating set of size k, but that a Hamiltonian tour of G' exists. Let D be the set of choices made by the tour in V_2, i.e. the set of vertices through which the tour exits V_2. The selection of the corresponding set in G leaves some vertex not dominated, say vertex i. Consider the gadget C_i. It can not be entered directly from a vertex in V_2, since none of the vertices from which we exited V_2 corresponds to one that dominates i. Also, if all the other gadgets are traversed in a way that does not exclude a Hamiltonian cycle, we established above that the set of entry indices in each is the same as the set of exit indices. Thus, if the set of input indices into a gadget C_j corresponds to its domination by some vertices in D, the tour when exiting C_j will be led to other gadgets also corresponding to dominated vertices, and therefore it will not be lead to C_i. Thus, we have a contradiction and no Hamiltonian tour is possible.

It remains to establish the converse, namely that a dominating set of size k implies a Hamiltonian tour. Let $D = \{d_1, d_2, \ldots, d_k\}$ be a dominating set. Let us first describe the tour outside the gadgets. Starting at u_1, move to v_{d_k+1} (once again, v_{n+1} is the same as v_1) and then follow the edges in V_2 until v_{d_1} is reached. Then we exit V_2 towards the gadgets. When we reach an *Out* gadget vertex that points to V_1 we move to u_2. From there we move to v_{d_1+1}, then to v_{d_2} and so on. This procedure makes sure that, even though we enter V_2 only k times, all of its n vertices are covered.

Now, suppose that our path has reached a gadget vertex $g_{(i,j,In)}$. This means that we are following an edge that "belongs" to the j-th vertex of G that could dominate vertex i. Call this vertex x. Take the first vertex in D that dominates i and comes lexicographically after x. Suppose that this is the j'-th vertex that can dominate i overall. (If there is no such vertex in D that dominates i and comes after x take the first vertex in D that dominates i). Now, the traversal of gadget C_i will be $g_{(i,j,In)} \to g_{(i,j+1,Out)} \to g_{(i,j+1,In)} \to \cdots \to g_{(i,j',Out)}$, from which point we exit the gadget. (Note that j and j' need not necessarily be distinct for the above argument to work).

Let us now prove this is indeed a Hamiltonian tour. It is not hard to see that all vertices of V_1 and V_2 are visited exactly once, leaving the gadgets as the hard part of this proof. The proof will be by induction. For gadget C_1 we know that it is entered directly from V_2 only (1 is always the first vertex lexicographically that any other vertex dominates). Observe that the tour we suggested is just a restatement of the reasoning we made on how gadgets work. Then it is not hard to see that, no matter which vertices of D dominate 1, C_1 will be traversed correctly. The problem is that paths have been now "shifted", i.e. when entering from the entry point of the i-th vertex that dominates 1 we exit from the exit point of the $(i+1)$-th vertex that dominates 1. Therefore, we will make a proof by induction. Suppose that we know that the tour we suggested visits the vertices of gadgets $C_1, \ldots, C_i$ exactly once and that the entry and exit points for each gadget correspond to the vertices of D that dominate the vertex this gadget represents. Now consider the gadget C_{i+1}. It is only entered using edges that "belong" to vertices of D, because of the inductive hypothesis. Suppose that one of its vertices is visited twice. Then, one of its entry points must be used twice, meaning that an exit of a previous gadget or a vertex of V_2 is visited twice, a contradiction. Now, suppose that the vertices of D that dominate vertex $i+1$ are the j_1-th, the j_2-th, and so on of the vertices that could dominate $i+1$ overall. Our tour must visit vertex $g_{(i+1,j_1,In)}$, because the vertex of some previous gadget that point to it is used as an output point (by the inductive hypothesis). Similar arguments hold for j_2 and so on, leading us to conclude that all appropriate entry points are used and gadget C_{i+1} is traversed correctly.

Finally, what is left is to argue that G' has low directed pathwidth. Consider the following cop strategy for the robber game: Keep k cops on the vertices of V_1 at all times. Now 2 cops are enough to clean V_2, since it is just a directed cycle with edges going out to V_3 but no other incoming edges. Then, these two cops can clean gadget C_1 for the same reasons. After that, they can clean C_2 and so on, until the whole graph is clean.

□

4 Max Di Cut

Let us now focus on a problem of much different nature: Max Di Cut. Even though, as we saw in Section 3, no digraph complexity measure manages to provide an FPT algorithm for Directed Hamiltonian Circuit, they do succeed in providing algorithms with polynomial running times, when the width k is fixed. For Max Di Cut the situation is much worse, as we will show that the problem is NP-hard even for $k = 1$. This creates an even larger gap with the FPT performance of treewidth than we had in the case of Directed Hamiltonian Circuit.

We will prove that Max Di Cut is both NP and APX-hard, even when restricted to DAGs by showing a reduction from the maximization version of NAE3SAT.

Theorem 2. Max Di Cut *is NP-hard and APX-hard, even when restricted to DAGs.*

Proof. We give a gap-preserving reduction from NAE3SAT to Max Di Cut.

Given a NAE3SAT formula ϕ with m clauses and n variables we construct a new NAE3SAT formula ϕ' with $2m$ clauses and n variables and show that ϕ is satisfiable iff ϕ' is satisfiable (satisfaction is in the NAESAT sense). Then from ϕ' we construct a (weighted) DAG G and show that ϕ' is satisfiable iff G has a directed cut of size $46m$. Without loss of generality, we may assume that every clause of ϕ has exactly three literals (otherwise we may repeat one).

The new formula ϕ' is constructed by taking ϕ and adding to it, for every clause $(l_1 \vee l_2 \vee l_3)$, the same clause with all literals complemented. If an assignment satisfies t clauses of the original formula, it must satisfy exactly $2t$ of the $2m$ clauses of ϕ'. Note that, if we denote by f_i the number of appearances of the variable x_i in ϕ, then the same variable will appear $2f_i$ times in ϕ': f_i times as x_i and f_i times as $\neg x_i$. In other words, the positive and negative appearances of each variable in ϕ' are balanced. We will make use of this fact several times.

Let us now construct the DAG $G(V, E)$. V consists of four disjoint sets of vertices A, X, C, B. $A = \{a_1, \ldots, a_n\}$ will be a set of source vertices. $B =$

$\{b_1, \ldots, b_{2m}\}$ will be a set of sink vertices. $X = \{x_1, x'_1, x_2, x'_2, \ldots, x_n, x'_n\}$ will be the set of vertices corresponding to literals of ϕ' while $C = \{c_{i,j,k} \mid i \in \{1, 2, \ldots, 2m\},\ j, k \in \{1, 2, 3\}\}$ will correspond to the clauses of ϕ'.

E consists of the following sets of weighted edges:

1. The set $E_1 = \{(a_i, x_i) \mid i \in \{1, \ldots, n\}\}$. Each of these edges has weight $6f_i$, where f_i is the total number of appearances of the variable x_i in ϕ.
2. The set $E_2 = \{(x_i, x'_i) \mid i \in \{1, \ldots, n\}\}$. Each of these edges also has weight $6f_i$.
3. The set $E_3 = \{(c_{i,j,k}, b_i) \mid i \in \{1, \ldots, 2m\},\ j, k \in \{1, 2, 3\}, j \neq k\}$. These have weight 1.
4. The set $E_4 = \{(c_{i,k,k}, c_{i,j,k} \mid i \in \{1, \ldots, 2m\},\ j, k \in \{1, 2, 3\}, j \neq k\}$. These also have weight 1.
5. Finally, we add edges that connect vertices of the set X to the corresponding vertices of C. That is, we add the edges $\{(x_l, c_{i,j,k}),\ k \in \{1, 2, 3\}\}$ when the literal x_l appears in the j-th position of the i-th clause of ϕ', and the edges $(x'_l, c_{i,j,k})$ when the literal $\neg x_l$ appears in that position. These edges have weight 2.

Suppose we are given a truth assignment that satisfies (in the NAESAT sense) t of the m clauses of ϕ. It must satisfy $2t$ of the $2m$ clauses of ϕ'. Let us partition V into V_0 and V_1. Place all vertices of A into V_0 and all vertices of B into V_1. Place the vertices of X that correspond to true literals in V_1 and the rest in V_0. Place the vertices of C that correspond to true literals in V_0 and the rest in V_1.

Let us calculate the weight of this cut. If a variable x_i is assigned the value 1 in the assignment, the edge (a_i, x_i) contributes $6f_i$ to the cut. If it is assigned 0, then x'_i is in V_1, therefore the edge (x_i, x'_i) contributes $6f_i$ to the cut. Thus, the total contribution of all edges in $E_1 \cup E_2$ is $6 \sum_i f_i$. Because the appearances of each variable in ϕ' are balanced, there are as many literals that took the value true as there are literals that took the value false, in any assignment. Therefore, exactly half the edges of E_3 contribute to the cut. The number of edges in E_3 is $12m$ so, a weight of $6m$ is contributed to the cut. It is not hard to see that, for a satisfied clause C_i, the edges of E_4 incident on vertices that correspond to this clause contribute exactly 2 to the cut. On the other hand, the edges of E_4 incident on vertices of C that correspond to a clause that is not satisfied will contribute 0 to the cut, since all these vertices correspond to literals with the same truth value and are therefore on the same side of the partition. Thus, we get a total of $4t$ contributed to the cut, since $2t$ clauses are satisfied. Finally, once again because of the balancing of ϕ', exactly half of the edges of E_5 contribute to the cut: those incident on vertices of X that we placed in V_0, i.e. vertices that correspond to false literals. Since the weight of each such edge is 2, this adds up to a total contribution of $2 \sum_i f_i$.

Thus, the total size of the cut is $6 \sum_i f_i + 6 \sum_i f_i + 2 \sum_i f_i + 4t$. But, since every clause of ϕ had exactly three literals, $\sum_i f_i = 3m$. Therefore, the weight of the cut is $42m + 4t$, which is equal to $46m$ when the truth assignment satisfies every clause of ϕ.

Now for the other direction, suppose we are given a partition of V into V_0 and V_1. We will show that we can transform such a cut into a cut of the previous form, thus obtaining a truth assignment. First, observe that for any optimal cut $A \subseteq V_0$ and $B \subseteq V_1$, because it is always optimal to place a source in V_0 and a sink in V_1. Now, suppose that in the cut we are given, for some i, $x_i, x'_i \in V_0$. Then place x'_i in V_1 and this will not make the cut smaller because now the edge (x_i, x'_i) contributes to the cut and its weight is exactly as much as the weight of all other edges incident on x'_i. Also, if $x_i, x'_i \in V_1$ place x_i in V_0. This can not make the cut smaller, since the only edge lost is (a_i, x_i) and its weight is the same as that of (x_i, x'_i) which now enters the cut. Therefore, we have now made sure that for all i, x_i and x'_i are on different sides of the partition, without decreasing the size of our cut.

Consider now a vertex $c_{i,j,j}$. We know that there exists an edge $(x_i, c_{i,j,j})$ (or an edge $(x'_i, c_{i,j,j})$) of weight 2, which is as much as the weight of all other edges incident on $c_{i,j,j}$. Therefore, if x_i (resp. x'_i) is in V_0, then we can place $c_{i,j,k}$ in V_1 without decreasing the size of the cut. Otherwise, we can place $c_{i,j,j}$ in V_0, because the edge of weight 2 can not be included in the cut by changing the side of $c_{i,j,j}$ only, and therefore placing it in V_0 is not worse because this way we may also include some of the other edges in the cut. This establishes that every vertex $c_{i,j,j}$ is on a different side of the partition from its predecessor in X.

Finally, consider a vertex $c_{i,j,k}$, $j \neq k$. If its predecessor in X is in V_0 we can place it in V_1 without decreasing the size of the cut, because then the edge of weight 2 is included. Otherwise we can place it in V_0, and this will include the edge $(c_{i,j,k}, b_i)$ in the cut. This does not decrease the size of the cut, since the edge of weight 2 was not included anyway, therefore we might at most lose the other edge incident on this vertex, which also has weight 1. This establishes that each of the remaining vertices of C is also on different a different side of the partition from its predecessor in X.

Now, observe that starting with any given cut, we have transformed it into a cut of a special form, without decreasing its size. From this cut we can construct a truth assignment: set to true the literals corresponding to vertices in X that we placed in V_1. This is a valid assignment, since exactly one of x_i, x'_i is in V_1. Also, if we repeat the process of the first direction of this reduction starting from this assignment we will get the same cut. Therefore, we have shown that there is a truth assignment that satisfies t of the m clauses of ϕ iff there is a cut in the DAG G of size at least $42m + 4t$. Thus,

$$OPT_{NAESAT}(\phi) = m \Rightarrow OPT_{MDC}(G) = 46m$$
$$OPT_{NAESAT}(\phi) < (1-\epsilon)m \Rightarrow OPT_{MDC}(G) \leq (1 - \frac{2\epsilon}{23})46m$$

□

It is not hard to extend the results of the previous theorem to the cardinality version of Max Di Cut, that is, the version where all edges have the same weight.

Theorem 3. *Cardinality* Max Di Cut *is NP and APX-hard, even when restricted to DAGs.*

Proof. First, observe that all the edge weights used in the proof of Theorem 2 are polynomially (in fact linearly) bounded by the size of the original NAE3SAT formula. Thus, if we extend the problem's definition to include multigraphs, we can replace every edge of weight w by w parallel edges of weight 1. It is not hard to see that this does not affect the rest of the proof.

Now, let us show how to eliminate parallel edges. For each edge (u, v) introduce a directed path of length 3 u, w_1, w_2, v where w_1 and w_2 are new vertices. Observe that, if u is assigned 0 and v is assigned 1, then it is possible to include 2 of the 3 edges of the path in the cut, by assigning 0 to w_2 and 1 to w_1. However, any other assignment to u and v ensures that at most 1 of the three edges can be included in the cut, and in fact this is always possible by assigning 0 to w_1 and 1 to w_2. Thus, it is not hard to see that the reduction's arguments can now be applied with little modification.

Corollary 1. Max Di Cut *is NP-hard and APX-hard even when restricted to graphs of bounded directed treewidth, DAG-width, kelly-width or directed pathwidth.*

Proof. The proof is immediate, because DAGs have width 1 under the definitions of all these widths.

5 Conclusions and Further Work

Discussion of results In this paper we have presented two hardness results affecting all known generalizations of treewidth to digraphs as well as directed pathwidth. It may be worthwhile at this point to discuss why such results hold for the directed cousins of treewidth, when in the undirected case there has been such a huge success.

First, the hardness result for Max Di Cut, gives us one indication why such hardness results hold. The reason is simply that for some problems DAGs are not really an "easy" topology, as trees are in the undirected case. Therefore, it would probably make sense in the future to focus research on directed treewidth variants on generalizations of a graph topology that is even simpler than a DAG. A further clue is given in this direction by the fact that DAGs (surprisingly) are the base case for both directed pathwidth and the three treewidth variants we considered. One would probably expect pathwidth and treewidth to be based on different graph topologies.

On the other hand, directed treewidth variants have had some success with path-based problems, such as Directed Hamiltonian Circuit. For such problems, DAGs usually are indeed the trivial case and it makes sense to design a width as a generalization of DAGs. However, we showed that none of the currently known widths (including directed pathwidth) is restrictive enough to provide for an FPT algorithm for Directed Hamiltonian Circuit.

Therefore, we believe that our results may suggest that in the directed case things may be more complicated and possibly no "right" complexity measure exists. On one hand, it would probably make sense to explore the possibility of a width (not based on DAGs) that can solve Max Di Cut and similar problems, while still being more general than undirected treewidth. And on the other hand, a more realistic goal might be to attempt to refine the definition of some of the already known widths (which are based on DAGs) in order to make it restrictive enough to solve Directed Hamiltonian Circuit and related path problems in FPT time.

References

1. János Barát. Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22(2):161–172, 2006.
2. Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In Igor Prívara and Peter Ruzicka, editors, *MFCS*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 1997.
3. Hans L. Bodlaender. Treewidth: Characterizations, applications, and computations. In Fedor V. Fomin, editor, *WG*, volume 4271 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.
4. Hans L. Bodlaender. Treewidth: Structure and algorithms. In Giuseppe Prencipe and Shmuel Zaks, editors, *SIROCCO*, volume 4474 of *Lecture Notes in Computer Science*, pages 11–25. Springer, 2007.
5. Bruno Courcelle. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.*, 85(1):12–75, 1990.
6. Rod G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, November 1999.
7. J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer, 1 edition, March 2006.
8. Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 637–644. SIAM, 2007.
9. Thor Johnson, Neil Robertson, Paul D. Seymour, and Robin Thomas. Directed tree-width. *J. Comb. Theory, Ser. B*, 82(1):138–154, 2001.
10. Rolf Niedermeier. *Invitation to Fixed Parameter Algorithms (Oxford Lecture Series in Mathematics and Its Applications)*. Oxford University Press, USA, March 2006.
11. Jan Obdrzálek. Dag-width: connectivity measure for directed graphs. In *SODA*, pages 814–821. ACM Press, 2006.
12. Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.
13. Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic Aspects of Tree-Width. *J. Algorithms*, 7(3):309–322, 1986.