Computing the largest bond of a graph

Gabriel L. Duarte  
Fluminense Federal University, RJ, Brazil  
gabriel@id.uff.br

Daniel Lokshtanov  
University of California Santa Barbara, CA, USA  
daniello@ucsb.edu

Lehilton L. C. Pedrosa  
University of Campinas, SP, Brazil  
lehilton@ic.unicamp.br

Rafael C. S. Schouery  
University of Campinas, SP, Brazil  
rafael@ic.unicamp.br

Uéverton S. Souza\textsuperscript{1}  
Fluminense Federal University, RJ, Brazil  
ueverton@ic.uff.br

---

Abstract

A bond of a graph \( G \) is an inclusion-wise minimal disconnecting set of \( G \), i.e., bonds are cut-sets that determine cuts \([S, V \setminus S]\) of \( G \) such that \( G[S] \) and \( G[V \setminus S] \) are both connected. Given \( s, t \in V(G) \), an \( st \)-bond of \( G \) is a bond whose removal disconnects \( s \) and \( t \). Contrasting with the large number of studies related to maximum cuts, there are very few results regarding the largest bond of general graphs. In this paper, we aim to reduce this gap on the complexity of computing the largest bond and the largest \( st \)-bond of a graph. Although cuts and bonds are similar, we remark that computing the largest bond of a graph tends to be harder than computing its maximum cut. We show that \textsc{Largest Bond} remains NP-hard even for planar bipartite graphs, and it does not admit a constant-factor approximation algorithm, unless \( P = NP \). We also show that \textsc{Largest Bond} and \textsc{Largest st-Bond} on graphs of clique-width \( w \) cannot be solved in time \( f(w) \times n^{O(w)} \) unless the Exponential Time Hypothesis fails, but they can be solved in time \( f(w) \times n^{O(w)} \). In addition, we show that both problems are fixed-parameter tractable when parameterized by the size of the solution, but they do not admit polynomial kernels unless \( NP \subseteq \text{coNP}/\text{poly} \).

2012 ACM Subject Classification  Mathematics of computing → Graph theory; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases  bond, cut, maximum cut, connected cut, FPT, treewidth, clique-width

Funding  Supported by Grant 2015/11937-9, São Paulo Research Foundation (FAPESP) and by Grant E-26/203.272/2017, Rio de Janeiro Research Foundation (FAPERJ) and by Grant 308689/2017-8, 425340/2016-3, 313026/2017-3, 422829/2018-8, 303726/2017-2, National Council for Scientific and Technological Development (CNPq).

Acknowledgements  We thank the organizers of WoPOCA 2017 for the opportunity to bring together some of the co-authors of this paper.

1 Introduction

Let \( G = (V, E) \) be a simple, connected, undirected graph. A disconnecting set of \( G \) is a set of edges \( F \subseteq E(G) \) whose removal disconnects \( G \). The edge-connectivity of \( G \) is

\[ \lambda(G) = \min \{ |F| : F \text{ is a disconnecting set of } G \} \]
Computing the largest bond of a graph

\( \kappa'(G) = \min\{|F| : F \text{ is a disconnecting set of } G\} \). A cut \([S, T]\) of \(G\) is a partition of \(V\) into two subsets \(S\) and \(T = V \setminus S\). The cut-set \(\partial(S)\) of a cut \([S, T]\) is the set of edges that have one endpoint in \(S\) and the other endpoint in \(T\); these edges are said to cross the cut.

In a connected graph, each cut-set determines a unique cut. Note that every cut-set is a disconnecting set, but the converse is not true. An inclusion-wise minimal disconnecting set of a graph is called a bond. It is easy to see that every bond is a cut-set, but there are cut-sets that are not bonds. More precisely, a nonempty set of edges \(F\) of \(G\) is a bond if and only if \(F\) determines a cut \([S, T]\) of \(G\) such that \(G[S]\) and \(G[T]\) are both connected. Let \(s, t \in V(G)\). An \(st\)-bond of \(G\) is a bond whose removal disconnects \(s\) and \(t\).

In this paper, we are interested in the complexity aspects of the following problem.

**LARGEST BOND**

**Instance:** A graph \(G = (V, E)\); a positive integer \(k\).

**Question:** Is there a proper subset \(S \subset V(G)\) such that \(G[S]\) and \(G[V \setminus S]\) are connected and \(|\partial(S)| \geq k\)?

We also consider LARGEST \(st\)-BOND, where given a graph \(G = (V, E)\), vertices \(s, t \in V(G)\), and a positive integer \(k\), we are asked whether \(G\) has an \(st\)-bond of size at least \(k\).

**LARGEST \(st\)-BOND**

**Instance:** A graph \(G = (V, E)\); vertices \(s, t \in V(G)\); a positive integer \(k\).

**Question:** Is there a proper subset \(S \subset V(G)\) with \(s \in S\) and \(t \notin S\), such that \(G[S]\) and \(G[V \setminus S]\) are connected and \(|\partial(S)| \geq k\)?

A minimum (maximum) cut of a graph \(G\) is a cut with cut-set of minimum (maximum) size. Every minimum cut is a bond, thus a minimum bond is also a minimum cut of \(G\), and it can be found in polynomial time using the classical Edmonds–Karp algorithm [12]. Besides that, minimum \(st\)-bonds are well-known structures, since they are precisely the \(st\)-cuts involved in the Gomory-Hu trees [21].

Regarding bonds on planar graphs, a folklore theorem states that if \(G\) is a connected planar graph, then a set of edges is a cycle in \(G\) if and only if it corresponds to a bond in the dual graph of \(G\) [19]. Note that each cycle separates the faces of \(G\) into the faces in the interior of the cycle and the faces of the exterior of the cycle, and the duals of the cycle edges are exactly the edges that cross from the interior to the exterior [30]. Consequently, the girth of a planar graph equals the edge connectivity of its dual [5].

Although cuts and bonds are similar, computing the largest bond of a graph seems to be harder than computing its maximum cut. MAXIMUM CUT is NP-hard in general [17], but becomes polynomial for planar graphs [22]. On the other hand, finding a longest cycle in a planar graph is NP-hard, implying that finding a largest bond of a planar multigraph (or of a simple edge-weighted planar graph) is NP-hard. In addition, it is well-known that if a simple planar graph is 3-vertex-connected, then its dual is a simple planar graph. In 1976, Garey, Johnson, and Tarjan [18] proved that the problem of establishing whether a 3-vertex-connected planar graph is Hamiltonian is NP-complete, thus, as also noted by Haglin and Venkatesan [23], finding the largest bond of a simple planar graph is also NP-hard, contrasting with the polynomial-time solvability of MAXIMUM CUT on planar graphs.

From the point of view of parameterized complexity, it is well known that MAXIMUM CUT can be solved in FPT time when parametrized by the size of the solution [27], and since every graph has a cut with at least half the edges [13], it follows that it has a linear kernel. Concerning approximation algorithms, a 1/2-approximation algorithm can be obtained by randomly partitioning the set vertices into two parts, which induces a cut-set whose expected size is at least half of the number of edges [28]. The best-known result is the...
Van Dyck, Brinkmann, Fack, and McKay [1] showed that if a Yutsis graph is regular with will have a largest bond with size at least $n$ with

verified the conjecture that any simple preprocessing the input in order to bound the treewidth of the resulting instance. After that, complexity of Largest Bond

Maximum Cut than an approximation algorithm, unless P fails. Moreover, we show that Largest Bond classes for which a general reduction that allows us to observe that Largest Bond of a graph. Preliminarily, we observe that while Largest Bond remains NP-hard for several classes for which Maximum Cut is NP-hard. Using this framework, we are able to show that Largest Bond on graphs of clique-width $w$ cannot be solved in time $f(w) \times n^{o(w)}$ unless the ETH fails. Moreover, we show that Largest Bond does not admit a constant-factor approximation algorithm, unless $P = NP$, and thus is asymptotically harder to approximate than Maximum Cut.

As for positive results, the main contributions of this work concern the parameterized complexity of Largest Bond. Using win/win approaches, we consider the strategy of preprocessing the input in order to bound the treewidth of the resulting instance. After that,
Computing the largest bond of a graph

by presenting a dynamic programming algorithm for \textsc{Largest Bond} parameterized by the treewidth, we show that the problem is fixed-parameter tractable when parameterized by the size of the solution. Finally, we remark that \textsc{Largest Bond} and \textsc{Largest st-Bond} do not admit polynomial kernels, unless \text{NP} \subseteq \text{coNP}/\text{poly}.

## 2 Intractability results

In this section, we discuss aspects of the hardness of computing the largest bond. Notice that \textsc{Largest Bond} is Turing reducible to \textsc{Largest st-Bond}. Therefore, the results presented in this section also hold for \textsc{Largest st-Bond}.

Although \textsc{Maximum Cut} is trivial for bipartite graphs, we first observe that the same does not apply to compute the largest bond.

▶ **Theorem 1.** \textsc{Largest Bond} is NP-complete for planar bipartite graphs.

**Proof.** It is well-known that a connected planar graph is Eulerian if and only if its dual graph is bipartite \cite{34}. In 1994, Picouleau \cite{31} proved that deciding whether a 4-regular planar graph has a Hamiltonian cycle is NP-complete. Thus, to determine the size of the largest bond of a planar bipartite multigraph is NP-complete. In order to obtain a simple planar bipartite graph, it is enough to subdivide each edge of the graph; notice that this operation preserves the size of the largest bond of the graph. Therefore, determining the size of the largest bond of a simple planar bipartite graph is NP-complete. ◀

▶ **Theorem 2.** Let $G$ be a simple bipartite graph and $\ell \in \mathbb{N}$. To determine the largest bond $\partial(S)$ of $G$ with $|S| = \ell$ is $W[1]$-hard with respect to $\ell$.

**Proof.** From an instance $H$ of \textsc{k-Independent Set} on regular graphs we first construct a multigraph $G'$ by adding an edge between any pair of vertices. Finally, we obtain a simple graph $G$ by subdividing every edge of $G'$. Notice that $H$ has an independent set of size $k$ if and only if $G$ has a bond $\partial(S)$ of size $dk + k(n - k)$ with $|S| = k + \binom{k}{2}$, where $d$ is the vertex degree of $H$. ◀

Next, we present a general framework for reducibility from \textsc{Maximum Cut} to \textsc{Largest Bond}, by defining a special graph operator $\psi$ such that \textsc{Maximum Cut} on a graph class $\mathcal{F}$ is reducible to \textsc{Largest Bond} on the image of $\mathcal{F}$ via $\psi$. An interesting particular case occurs when $\mathcal{F}$ is closed under $\psi$ (for instance, chordal graphs are closed under $\psi$).

▶ **Definition 3.** Let $G$ be a graph and let $n = V(G)$. The graph $\psi(G)$ is constructed as follows:
(i) create $n$ disjoint copies $G_1, G_2, \ldots, G_n$ of $G$;
(ii) add vertices $v_a$ and $v_b$;
(iii) add an edge between $v_a$ and $v_b$;
(iv) add all possible edges between $V(G_1 \cup G_2 \cup \ldots \cup G_n)$ and $\{v_a, v_b\}$.

▶ **Definition 4.** A set of graphs $\mathcal{G}$ is closed under operator $\psi$ if whenever $G \in \mathcal{G}$, then $\psi(G) \in \mathcal{G}$.

▶ **Theorem 5.** \textsc{Largest Bond} is NP-complete for any graph class $\mathcal{G}$ such that:
- $\mathcal{G}$ is closed under operator $\psi$; and
- \textsc{MaxCut} is NP-complete for graphs in $\mathcal{G}$.
Proof. Let $G \in \mathcal{G}$, $n = |V(G)|$, and $H = \psi(G)$. By (i), $H \in \mathcal{G}$. Suppose $G$ has a cut $\{S, V(G) \setminus S\}$ of size $k$, and let $S_1, S_2, \ldots, S_n$ be the copies of $S$ in $G_1, G_2, \ldots, G_n$, respectively. If $S' = \{v_a\} \cup S_1 \cup S_2 \cup \ldots \cup S_n$, then $[S', V(H) \setminus S']$ defines a bond $\partial(S')$ of $H$ of size at least $nk + n^2 + 1$. Conversely, suppose $H$ has a bond $\partial(S')$ of size at least $nk + n^2 + 1$. We consider the following cases: (a) If $\{v_a, v_b\} \subseteq S'$, then for all copies $G_i$ but one we have $V(G_i) \subseteq S'$, as otherwise the graph induced by $V(H) \setminus S'$ would not be connected, and $\partial(S')$ would not be a bond. Thus, $V(H) \setminus S' \subseteq V(G_j)$ for some $j$, then the size of $\partial(S')$ is smaller than $nk + n^2 + 1$, a contradiction. (b) If $v_a \in S'$ and $v_b \notin S'$, then $\{v_a, v_b\}$ is incident with exactly $n^2 + 1$ edges crossing $[S', V(H) \setminus S']$, which implies that at least one copy $G_i$ has $k$ or more edges crossing $[S', V(H) \setminus S']$. Therefore, $G$ has a cut of size at least $k$.

\begin{corollary}
LARGEST BOND is NP-complete for the following classes:
1. chordal graphs;
2. co-comparability graphs;
3. $P_5$-free graphs.
\end{corollary}

Proof. Bodlaender and Jansen proved that MAXIMUM CUT is NP-complete when restricted to split and co-bipartite graphs. Since split graphs are chordal and co-bipartite graphs are $P_5$-free and co-comparability graphs, the NP-completeness also holds for these classes.

Now we have to show that the classes are closed under $\psi$.

(i) A graph is chordal if every cycle of length at least 4 has a chord. Let $G$ be a chordal graph. Notice that the disjoint union of $G_1, G_2, \ldots, G_n$ is also chordal. In addition, no chordless cycle of length at least 4 may contain either $v_a$ or $v_b$ because both vertices are universal. Therefore, $\psi(G)$ is chordal.

(ii) A graph is a co-comparability graph if it is the intersection graph of curves from a line to a parallel line. Let $G$ be a co-comparability graph. Notice that the class of co-comparability graphs is closed under disjoint union. Thus, in order to conclude that $\psi(G)$ is co-comparability, it is enough to observe that from a representation of curves (from a line to a parallel line) of the disjoint union of $G_1, G_2, \ldots, G_n$, one can construct a representation of $\psi(G)$ by adding two concurrent lines (representing $v_a$ and $v_b$) crossing all curves.

(iii) The disjoint union of $P_5$-free graphs is also $P_5$-free. In addition, no induced $P_5$ contains either $v_a$ or $v_b$ because both vertices are universal. Then, the class of $P_5$-free graphs is closed under $\psi$.

\subsection{Algorithmic lower bound for clique-width parameterization}

The CLIQUE-WIDTH of a graph $G$, denoted by $cw(G)$, is defined as the minimum number of labels needed to construct $G$, using the following four operations:

1. Create a single vertex $v$ with an integer label $\ell$ (denoted by $\ell(v)$);
2. Take the disjoint union (i.e., co-join) of two graphs (denoted by $\oplus$);
3. Join by an edge every vertex labeled $i$ to every vertex labeled $j$ for $i \neq j$ (denoted by $\eta(i,j)$);
4. Relabel all vertices with label $i$ by label $j$ (denoted by $\rho(i,j)$).

An algebraic term that represents such a construction of $G$ and uses at most $w$ labels is said to be a \textit{w-expression} of $G$, and the clique-width of $G$ is the minimum $w$ for which $G$ has a $w$-expression.
Computing the largest bond of a graph

Graph classes with bounded clique-width include cographs, distance-hereditary graphs, graphs of bounded treewidth, graphs of bounded branchwidth, and graphs of bounded rank-width.

In the '90s, Courcelle, Makowsky, and Rotics [7] proved that all problems expressible in MS1-logic are fixed-parameter tractable when parameterized by the clique-width of a graph and the logical expression size. The applicability of this meta-theorem has made clique-width become one of the most studied parameters in parameterized complexity. However, although several problems are MS1-expressible, this is not the case with \textsc{Maximum Cut}.

In 2014, Fomin, Golovach, Lokshtanov and Saurabh [13] showed that \textsc{Maximum Cut} on a graph of clique-width \( w \) cannot be solved in time \( f(w) \times n^o(w) \) for any function \( f \) of \( w \) unless Exponential Time Hypothesis (ETH) fails. Using operator \( \psi \), we are able to extend this result to \textsc{Largest Bond}.

\begin{lemma}
\textbf{Largest Bond} on graphs of clique-width \( w \) cannot be solved in time \( f(w) \times n^o(w) \) unless the ETH fails.
\end{lemma}

\begin{proof}
\textsc{Maximum Cut} cannot be solved in time \( f(w) \times n^o(w) \) on graphs of clique-width \( w \), unless Exponential Time Hypothesis (ETH) fails [13]. Therefore, by the polynomial-time reduction presented in Theorem 5 it is enough to show that the clique-width of \( \psi(G) \) is upper bounded by a linear function of the clique-width of \( G \).

If \( G \) has clique-width \( w \), then the disjoint union \( H_1 = G_1 \oplus G_2 \oplus \ldots \oplus G_n \) has clique-width \( w \). Suppose that all vertices in \( H_1 \) have label 1. Now, let \( H_2 \) be the graph isomorphic to a \( K_2 \) such that \( V(H) = \{v_a, v_b\} \), and \( v_a, v_b \) are labeled with 2. In order to construct \( \psi(G) \) from \( H_1 \oplus H_2 \) it is enough to apply the join \( \eta(1, 2) \). Thus, \( \psi(G) \) has clique-width equals \( w \).
\end{proof}

2.2 \hspace{1em} Inapproximability

While the maximum cut of a graph has at least a constant fraction of the edges, the size of the largest bond can be arbitrarily smaller than the number of edges; take, e.g., a cycle on \( n \) edges, for which a largest bond has size 2. This discrepancy is also reflected on the approximability of the problems. Indeed, we show that \textsc{Largest Bond} is strictly harder to approximate than \textsc{Maximum Cut}. To simplify the presentation, we consider a weighted version of the problem in which edges are allowed to have weights 0 or 1; the hardness results will follow for the unweighted case as well. In the \textsc{Binary Weighted Largest Bond}, the input is given by a connected weighted graph \( H \) with weights \( w : E(H) \to \{0, 1\} \). The objective is to find a bond whose total weight is maximum.

Let \( G \) be a graph on \( n \) vertices and whose maximum cut has size \( k \). Next, we define the \( G \)-edge embedding operator \( \xi_G \). Given a connected weighted graph \( H \), the weighted graph \( \xi_G(H) \) is constructed by replacing each edge \( \{u, v\} \in E(H) \) with weight 1 by a copy of \( G \), denoted by \( G_{uv} \), whose edges have weight 1, and, for each vertex \( t \) of \( G_{uv} \), new edges \( \{u, t\} \) and \( \{v, t\} \), both with weight 0.

We can also apply the \( G \)-edge embedding operation on the graph \( \xi_G(H) \), and so on. In what follows, for an integer \( h \geq 0 \), denote by \( \xi_G^h(H) \) the graph resulting from the operation that receives a graph \( H \) and applies \( \xi_G \) successively \( h \) times. Notice that \( \xi_G^h(H) \) can be constructed in \( O(|V(G)|^{h+1}) \) time. For some \( j \), \( 0 \leq j \leq h - 1 \), observe that an edge \( \{u, v\} \in E(\xi_G^j(H)) \) will be replaced by a series of vertices added in iterations \( j + 1, j + 2, \ldots, h \). These vertices will be called the descendants of \( \{u, v\} \), and will be denoted by \( V_{uv} \).

Let \( K_2 \) be the graph composed of a single edge \( \{u, v\} \), and consider the problem of finding a bond of \( \xi_G(K_2) \) with maximum weight. Since edges connecting \( u \) or \( v \) have weight 0, one
can assume that $u$ and $v$ are in different sides of the bond, and the problem reduces to finding a maximum cut of $G$. In other words, the operator $\xi_G$ embeds an instance $G$ of MAXIMUM CUT into an edge $\{u, v\}$ of $K_2$.

This suggests the following strategy to solve an instance of MAXIMUM CUT. For some constant integer $h \geq 1$, calculate $H = \xi^h_G(K_2)$, and obtain a bond $F$ of $H$ with maximum weight. Note that, to solve $H$, one must solve embedded instances of MAXIMUM CUT in multiple levels simultaneously. For a level $j$, $1 \leq j \leq h - 1$, each edge $\{u, v\} \in E(\xi^j_G(K_2))$ with weight 1 will be replaced by a graph $G_{uv}$ which is isomorphic to $G$. In Lemma 9 below, we argue that $F$ is such that either $V(G_{uv}) \cup \{u, v\}$ are all in the same side of the cut, or $u$ and $v$ are in distinct sides. In the latter case, the edges of $F$ that separate $u$ and $v$ will induce a cut of $G$.

In the remaining of this section, we consider a constant integer $h \geq 0$. Then, we define $H^j = \xi^j_G(K_2)$ for every $j$, $0 \leq j \leq h$, and $H = H^h$. Also, we write $[S,T]$ to denote the cut induced by a bond $F$ of $H$.

**Definition 8.** Let $F$ be a bond of $H$ with cut $[S,T]$. We say that an edge $\{u,v\} \in E(H^j)$ with weight 1 is nice for $F$ if either

- $\{u,v\} \cap S = 1$, or
- $\{u,v\} \cup V_{uv} \subseteq S$, or
- $\{u,v\} \cup V_{uv} \subseteq T$.

Also, we say that $F$ is nice if, for every $j$, $0 \leq j \leq h - 1$, and every edge $\{u,v\} \in E(H^j)$ with weight 1, $\{u,v\}$ is nice for $F$.

**Lemma 9.** There is a polynomial-time algorithm that receives a bond $F$, and finds a nice bond $F'$ such that $w(F') = w(F)$.

**Proof.** Let $[S,T]$ be the cut induced by $F$ and let $j^*$ be minimum such that there exists an edge $\{u,v\} \in H^{j^*}$ with weight 1 which is not nice for $F$. Then $\{u,v\} \cap S \neq 1$. Assume, without loss of generality that $u,v \in S$. In this case, $U := V_{uv} \cap T$ is not empty. Since removing vertices $\{u,v\}$ disconnects $U$, and $T$ must be connected, it follows that $U = T$.

This implies that $N(T) \subseteq (V_{uv} \setminus T) \cup \{u,v\}$.

We will construct a bond $F'$ of $H$ with cut $[S',T']$. Let $S'$ be the set of vertices in the connected component of $H[S \setminus \{v\}]$ which contains $u$, and $T' = V(H) \setminus S'$. Since $H[S]$ is connected, so must be $H[S \setminus S']$. Also, each vertex of $U$ is adjacent to $v$, thus $H([S \setminus S'] \cup U)$ is connected. Observe that $T' = (S \setminus S') \cup U$, so indeed the cut $[S',T']$ induces a bond $F' = \emptyset(S')$. Observe that any edge that appears only in $F$ or only in $F'$ is adjacent to $v$. Since such edges have weight 0, this implies $w(F) = w(F')$.

To complete the proof, we claim that if for some $j$, $0 \leq j \leq h$, there exists an edge $\{u,v\} \in H^j$ with weight 1 which is not nice for $F'$, then $j > j^*$. If this claim holds, then we need to repeat the previous procedure at most $h$ times before obtaining a nice bond $F'$.

To prove the claim, consider an edge $\{s,t\} \in H^j$ which is not nice for $F'$. Suppose, for a contradiction, that $V_{st} \cap V_{uv} = \emptyset$. There are two possibilities. If $s,t \in S'$, then $V_{st} \subseteq S'$; if $s,t \in T'$, then $V_{st} \subseteq S \setminus S' \subseteq T'$. In either possibility, $\{s,t\}$ is nice for $F'$. This is a contradiction, and thus $V_{uv} \cap V_{st} \neq \emptyset$.

The statement $V_{uv} \cap V_{st} \neq \emptyset$ can only happen if $V_{uv} \subseteq V_{st}$ or $V_{st} \subseteq V_{uv}$. If $V_{uv} \subseteq V_{st}$, then $U \subseteq V_{st}$ and $s,t \in S$. This implies that $\{s,t\}$ is not nice for $F$. But in this case $j < j^*$, contradicting the choice of $j^*$. Therefore, $V_{st} \subseteq V_{uv}$, and $j > j^*$, proving our claim.

In the following, assume that $F$ is a nice bond with cut $[S,T]$. Consider a level $j$, $0 \leq j \leq h$, and an edge $\{u,v\} \in E(H^j)$ with weight 1 such that $|\{u,v\} \cap S| = 1$. If $j < h$,
Computing the largest bond of a graph

then we define $F_{uw}$ to be the subset of edges in $F$ which are incident with some vertex of $V_{uw}$; if $j = h$, then we define $F_{uw} = \{\{u, v\}\}$. Note that, because $F$ is nice, if $|\{u, v\} \cap S| \neq 1$, then no edge of $F$ is incident with $V_{uw}$.

Suppose now that $|\{u, v\} \cap S| = 1$ for some edge $\{u, v\} \in E(H^j)$ with weight $1$ and $0 \leq j \leq h - 1$. In this case, $F$ induces a cut-set of $G_{uw}$. Namely, define $\hat{S}_{uw} := S \cap V(G_{uw})$ and $\hat{T}_{uw} := T \cap V(G_{uw})$ and let $\hat{F}_{uw}$ be the cut-set of $G_{uw}$ corresponding to cut $[\hat{S}_{uw}, \hat{T}_{uw}]$.

Observe that for distinct edges $\{u, v\}$ and $\{s, t\}$, it is possible that $|\hat{F}_{uv}| \neq |\hat{F}_{st}|$. We will consider bonds $F$ for which all induced cut-sets $\hat{F}_{uv}$ have the same size.

**Definition 10.** Let $\ell$ be a positive integer. A bond $F$ of $H$ with cut $[S, T]$ is said to be $\ell$-uniform if, (i) $F$ is nice, and (ii) for every $j$, $0 \leq j \leq h - 1$, and every edge $\{u, v\} \in E(H^j)$ with weight $1$ such that $|\{u, v\} \cap S| = 1$, $|\hat{F}_{uv}| = \ell$.

An $\ell$-uniform bond induces a cut-set of $G$ of size $\ell$.

**Lemma 11.** Suppose $F$ is an $\ell$-uniform bond of $H$. One can find in polynomial time a cut-set $L$ of $G$ with $|L| = \ell$.

**Proof.** Let $u, v$ be the vertices of $K_2$ to which $\xi_G$ was applied. Since $F$ is $\ell$-uniform, $|\hat{F}_{uv}| = \ell$. Note that $\hat{F}_{uv}$ induces a cut-set of size $\ell$ on $G$. ▶

In the opposite direction, a cut of $G$ induces an $\ell$-uniform bond of $H$.

**Lemma 12.** Suppose $L$ is a cut-set of $G$ with $|L| = \ell$. One can find in polynomial time an $\ell$-uniform bond $F$ of $H$ with $w(F) = \ell^h$.

**Proof.** For each $j \geq 0$, we construct a bond $F^j$ of $H^j$. For $j = 0$, let $F^0$ be the set containing the unique edge of $H^0 = K_2$. Suppose now that we already constructed a bond $F^{j-1}$ of $H^{j-1}$. For each edge $\{u, v\} \in F^{j-1}$, let $L_{uv}$ be the set of edges of $G_{uw}$ corresponding to $L$. Define $F^j := \cup_{\{u, v\} \in F^{j-1}} L_{uv}$. One can verify that indeed $F^j$ is a bond of $H^j$, and that $w(F_j) = |L| \times w(F_{j-1}) = \emptyset$. ▶

**Lemma 13.** There is a polynomial-time algorithm that receives a bond $F$ of $H$, and finds an $\ell$-uniform bond $F'$ of $H$ such that $w(F') = \ell^h \geq w(F)$.

**Proof.** Let $[S, T]$ be the cut corresponding to $F$. First, find the largest cut-set of a graph $G_{uw}$ over cut-sets $\hat{F}_{uv}$. More precisely, define $\hat{F}$ to be the cut-set $\hat{F}_{uv}$ with maximum $|\hat{F}_{uv}|$ over all edges $\{u, v\} \in E(H^j)$ with weight $1$ such that $|\{u, v\} \cap S| = 1$, and over all $j$, $0 \leq j \leq h - 1$. Let $\hat{F} := |\hat{F}|$.

We claim that for every $j$, $0 \leq j \leq h$, and every edge $\{u, v\} \in E(H^j)$ with weight $1$ such that $|\{u, v\} \cap S| = 1$, $w(F_{uv}) \leq \ell^h - j$. The proof is by (backward) induction on $j$. For $j = h$, $F_{uv} = \{u, v\}$, so $w(F_{uv}) = 1$. Next, let $j < h$, and assume the claim holds for $j + 1$.

Let $F^0_{uv}$ be the subset of edges in $F_{uv}$ incident with $u$ or $v$. The set $F_{uv}$ can be partitioned into $F^0_{uv}$ and sets $F_{st}$ for $\{s, t\} \in \hat{F}_{uv}$. To see this, observe that each edge $\{x, y\} \in F_{uv}$ must be incident with the vertices of $V_{st}$, and thus $\{x, y\}$ is incident with vertices of $V_{st}$ for some edge $\{s, t\} \in E(G_{uw})$. Since $|\{x, y\} \cap S| = 1$, neither $V_{st} \cup \{s, t\} \subseteq S$, nor $V_{st} \cup \{s, t\} \subseteq T$. Because $F$ is nice, it follows that $|\{s, t\} \cap S| = 1$, then $\{s, t\} \in \hat{F}_{uv}$, and thus $\{x, y\} \in F_{st}$. To complete the claim, observe that, by the induction hypothesis, $w(F_{st}) \leq \ell^h - j - 1$ for each $\{s, t\} \in \hat{F}_{uv}$, and recall that $|\hat{F}_{uv}| \leq |\hat{F}|$. Therefore

$$w(F) = w(F^0_{uv}) + \sum_{\{s, t\} \in \hat{F}_{uv}} w(F_{st}) \leq |\hat{F}| \times \ell^h - j - 1 = \ell^h - j.$$  

Using Lemma [12] for $\hat{F}$, we construct a bond $F'$ for $H$ with $w(F') = \ell^h$. ▶
Lemma 14. Let $F^*$ be a bond of $H$ with maximum weight. Then $w(F^*) = k^h$.

Proof. We assume that $F^*$ is $\ell$-uniform such that $w(F^*) = \ell^h$ for some $\ell$; if this is not the case, then use Lemma 13.

Since $F^*$ is $\ell$-uniform, using Lemma 12 one obtains a cut-set $L$ of $G$ with size $\ell$, then $\ell \leq k$, and thus $w(F^*) \leq k^h$.

Conversely, let $L$ be a cut-set of $G$ with size $k$. Using Lemma 12 for $L$, we obtain a bond $F$ of $H$ with weight $k^h$, and thus $w(F^*) \geq k^h$.

Lemma 15. If there exists a constant-factor approximation algorithm for Weighted Largest Bond, then $P = \text{NP}$.

Proof. Consider a graph $G$ whose maximum cut has size $k$. Construct graph $H$ and obtain a bond $F$ of $H$ using an $\alpha$-approximation, for some constant $0 < \alpha < 1$. Using the algorithm of Lemma 13, obtain an $\ell$-uniform bond $F'$ of $H$ such that $w(F') = \ell^h \geq w(F)$. Using Lemma 14 and the fact that $F'$ is an $\alpha$-approximation, $\ell^h \geq \alpha \times k^h$. Using Lemma 11 one can obtain a cut-set $L$ of $G$ with size $\ell \geq \alpha^\frac{1}{h}k$.

For any constant $\varepsilon$, $0 < \varepsilon < 1$, we can set $h = \lceil \log_{1-\varepsilon} \alpha \rceil$, such that the cut-set $L$ has size at least $\ell \geq (1-\varepsilon)k$. Since MAXIMUM CUT is APX-hard, this implies $P = \text{NP}$.

Theorem 16. If there exists a constant-factor approximation algorithm for Largest Bond, then $P = \text{NP}$.

Proof. We show that if there exists an $\alpha$-approximation algorithm for LARGEST BOND, for constant $0 < \alpha < 1$, then there is an $\alpha/2$-approximation algorithm for the Binary Weighted Largest Bond, so the theorem will follow from Lemma 15.

Let $H$ be a weighted graph whose edge weights are all 0 or 1. Let $m$ be the number of edges with weight 0, and let $l$ be the weight of a bond of $H$ with maximum weight. Assume $l \geq 2/\alpha$, as otherwise, one can find an optimal solution in polynomial time by enumerating sets of up to $2/\alpha$ edges.

Construct an unweighted graph $G$ as follows. Start with a copy of $H$ and, for each edge $\{u,v\} \in E(H)$ with weight 1, replace $\{u,v\} \in E(G)$ by $m$ parallel edges. Finally, to obtain a simple graph, subdivide each edge of $G$. If $F$ is a bond of $G$, then one can construct a bond $F'$ of $H$ by undoing the subdivision and removing the parallel edges. Each edge of $F'$ has weight 1, with exception of at most $m$ edges. Thus, $w(F') \geq (|F| - m)/m$.

Observe that an optimal bond of $H$ induces a bond of $G$ with size at least $ml$. Thus, if $F$ is an $\alpha$-approximation for $G$, then $|F| \geq \alpha ml$ and therefore

$$w(F') \geq \frac{\alpha ml - m}{m} = \alpha l - 1 \geq \alpha l - \alpha l/2 = \alpha l/2.$$

We conclude that $F'$ is an $\alpha/2$-approximation for $H$.

3 Algorithmic upper bounds for clique-width parameterization

Lemma 7 shows that LARGEST BOND on graphs of clique-width $w$ cannot be solved in time $f(w) \times n^{\omega(w)}$ unless the ETH fails. Now, we show that given an expression tree of width at most $w$, LARGEST BOND can be solved in $f(w) \times n^{O(w)}$ time.

An expression tree $T$ is irredundant if for any join node $\eta(i,j)$, the vertices labeled by $i$ and $j$ are not adjacent in the graph associated with its child. It was shown by Courcelle and Olariu [4] that every expression tree $T$ of $G$ can be transformed into an irredundant
Computing the largest bond of a graph

expression tree $T$ of the same width in time linear in the size of $T$. Therefore, without loss of generality, we can assume that $T$ is irredundant.

Our algorithm is based on dynamic programming over the expression tree of the input graph. We first describe what we store in the tables corresponding to the nodes in the expression tree.

Given a $w$-labeled graph $G$, two connected components of $G$ has the same type if they have the same set of labels. Thus, a $w$-labeled graph $G$ has at most $2^w - 1$ types of connected components.

Now, for every node $X_{\ell}$ of $T$, denote by $G_{X_{\ell}}$ the $w$-labeled graph associated with this node, and let $L_1(X_{\ell}), \ldots, L_w(X_{\ell})$ be the sets of vertices of $G_{X_{\ell}}$ labeled with $1, \ldots, w$, respectively. We define a table where each entry is of the form $c[i, s_1, \ldots, s_w, r, e_1, \ldots, e_{2w-1}, d_1, \ldots, d_{2w-1}]$, such that: $0 \leq s_i \leq |L_i(X_{\ell})|$ for $1 \leq i \leq w$; $0 \leq r \leq |E(G_{X_{\ell}})|$; $0 \leq e_i \leq \min(2, |L_i(X_{\ell})|)$ for $1 \leq i \leq 2^w - 1$; and $0 \leq d_i \leq \min(2, |L_i(X_{\ell})|)$ for $1 \leq i \leq 2^w - 1$.

Each entry of the table represents whether there is a partition $V_1, V_2$ of $V(G_{X_{\ell}})$ such that: $|V_1 \cap L_i(G_{X_{\ell}})| = s_i$; the cut-set of $[V_1, V_2]$ has size at least $r$; $G_{X_{\ell}}[V_1]$ has $e_i$ connected components of type $i$; $G_{X_{\ell}}[V_2]$ has $d_i$ connected components of type $i$, where $e_i = 2$ means that $G_{X_{\ell}}[V_1]$ has at least two connected components of type $i$. The same holds for $d_i$.

Notice that this table contains $f(w) \times n^{O(w)}$ entries. If $X_{\ell}$ is the root node of $T$ (that is, $G = G_{X_{\ell}}$), then the size of the largest bond of $G$ is equal to the maximum value of $r$ for which the table for $X_{\ell}$ contains a valid entry (true value), such that there are $j$ and $k$ such that $e_j = 0, e_j = 1$ for $1 \leq i, j \leq 2^w - 1, i \neq j$; and $d_k = 1, d_k = 1$ for $1 \leq i, k \leq 2^w - 1, i \neq k$.

It is easy to see that we store enough information to compute a largest bond. Note that a $w$-labeled graph is connected if and only if it has exactly one type of connected components and exactly one component of such a type.

Now we provide the details of how to construct and update such tables. The construction for introduce nodes of $T$ is straightforward.

**Relabel node:** Suppose that $X_{\ell}$ is a relabel node $\rho(i, j)$, and let $X_{p}$ be the child of $X_{\ell}$. Then the table for $X_{\ell}$ contains a valid entry $c[i, s_1, \ldots, s_w, r, e_1, \ldots, e_{2w-1}, d_1, \ldots, d_{2w-1}]$ if and only if the table for $X_{p}$ contains an entry $c[i', s_1', \ldots, s_w', r, e_1', \ldots, e_{2w-1}', d_1', \ldots, d_{2w-1}'] = true$, where: $s_i = 0; s_j = s_i' + s_j', s_p = s_p'$ for $1 \leq p \leq w, p \neq i, j$; $e_p = e_p'$ for any type that contains neither $i$ nor $j$; $e_p = 0$ for any type that contains $i$; and for any type $e_p$ that contains $j$, it holds that $e_p = \min(2, e_p' + e_i' + e_j')$ where $e_i'$ represent the set of labels $(C_p \setminus \{j\}) \cup \{i\}$, $e_j'$ represent the set of labels $C_p \cup \{i\}$, and $C_p$ is the set of labels associated to $p$. The same holds for $d_1', \ldots, d_{2w-1}'$.

**Union node:** Suppose that $X_{\ell}$ is a union node with children $X_p$ and $X_{p'}$. It holds that $c[i, s_1, \ldots, s_w, r, e_1, \ldots, e_{2w-1}, d_1, \ldots, d_{2w-1}]$ equals true if and only if there are valid entries $c[i', s_1', \ldots, s_w', r', e_1', \ldots, e_{2w-1}', d_1', \ldots, d_{2w-1}']$ and $c[i'', s_1'', \ldots, s_w'', r'', e_1'', \ldots, e_{2w-1}'', d_1'', \ldots, d_{2w-1}'']$, having: $s_i = s_i' + s_i''$ for $1 \leq i \leq w; r' + r'' \geq r; e_k = \min(2, e_k' + e_k''),$ and $d_k = \min(2, d_k' + d_k'')$ for $1 \leq k \leq 2^w - 1$.

**Join node:** Finally, let $X_{\ell}$ be a join node $\eta(i, j)$ with the child $X_p$. Remind that since the expression tree is irredundant then the vertices labeled by $i$ and $j$ are not adjacent in the graph $G_{X_p}$. Therefore, the entry $c[i, s_1, \ldots, s_w, r, e_1, \ldots, e_{2w-1}, d_1, \ldots, d_{2w-1}]$ equals true if and only if there is a valid entry $c[i', s_1', \ldots, s_w, r', e_1', \ldots, e_{2w-1}', d_1', \ldots, d_{2w-1}']$ where

$$r' + s_i \times (|L_j(X_p)| - s_j) + s_j \times (|L_i(X_p)| - s_i) \geq r,$$

and $e_p = e_p'$, case $p$ is associated to a type that contains neither $i$ nor $j$; $e_p = 1$, case $p$ is associated to $C_{i,j}^\ell \setminus \{i\}$, where $C_{i,j}^\ell$ is the set of labels obtained by the union of the types.
of $G_{X,\ell}$, with some connected component having either label $i$ or label $j$; $c_p = 0$, otherwise. The same holds for $d_1, \ldots, d_{2w-1}$.

The correctness of the algorithm follows from the description of the procedure. Since for each $\ell$, there are $O((n+1)^w \times m \times (3^{2w-1})^2)$ entries, the running time of the algorithm is $f(w) \times n^{O(w)}$. This algorithm together with Lemma 7 concludes the proof of Theorem 17.

▶ Theorem 17. Largest Bond cannot be solved in time $f(w) \times n^{O(w)}$ unless ETH fails, where $w$ is the clique-width of the input graph. Moreover, given an expression tree of width at most $w$, Largest Bond can be solved in time $f(w) \times n^{O(w)}$.

In order to extend this result to Largest st-Bond, it is enough to observe that given a tree expression $T$ of $G$ with width $w$, it is easy to construct a tree expression $T'$ with width equals $w + 2$, where no vertex of $V(G)$ has the same label than either $s$ or $t$. Let $w + 1$ be the label of $s$, and let $w + 2$ be the label of $t$. By fixing, for each $\ell$, $s_{w+1} = |L_{w+1}(X_\ell)|$ and $s_{w+2} = 0$, one can solve Largest st-Bond in time $f(w) \times n^{O(w)}$.

4 Bounding the treewidth of $G$

In the remainder of this paper we deal with our main problems: Largest Bond and Largest st-Bond parameterized by the size of the solution ($k$). Inspired by the principle of preprocessing the input to obtain a kernel, we consider the strategy of preprocessing the input in order to bound the treewidth of the resulting instance.

We start our analysis with Largest Bond.

▶ Definition 18. A graph $H$ is called a minor of a graph $G$ if $H$ can be formed from $G$ by deleting edges, deleting vertices, and by contracting edges. For each vertex $v$ of $H$, the set of vertices of $G$ that are contracted into $v$ is called a branch set of $H$.

▶ Lemma 19. Let $G$ be a simple connected undirected graph, and $k$ be a positive integer. If $G$ contains $K_{2,k}$ as a minor then $G$ has a bond of size at least $k$.

Proof. Let $H$ be a minor of $G$ isomorphic to $K_{2,k}$. Since $G$ is connected and each branch set of $H$ induces a connected subgraph of $G$, from $H$ it is easy to construct a bond of $G$ of size at least $k$. ◀

Combined with Lemma 19 the following results show that, without loss of generality, our study on $k$-bonds can be reduced to graphs of treewidth $\mathcal{O}(k)$.

▶ Lemma 20. Every graph $G = (V,E)$ contains $K_{2,k}$ as a minor or has treewidth at most $2k - 2$.

▶ Lemma 21. There is a polynomial-time algorithm that either concludes that the input graph $G$ contains $K_{2,k}$ as a minor, or outputs a tree-decomposition of $G$ of width at most $2k - 2$.

From Lemma 19 and Lemma 21 it follows that there is a polynomial-time algorithm that either concludes that the input graph $G$ has a bond of size at least $k$, or outputs a tree-decomposition of $G$ of width at most $2k - 2$. 

4.1 The st-bond case

Let $S \subseteq V(G)$ and let $\partial(S)$ be a bond of a connected graph $G$. Recall that a block is a 2-vertex-connected subgraph of $G$ which is inclusion-wise maximal, and a block-cut tree of $G$ is a tree whose vertices represent the blocks and the cut vertices of $G$, and there is an edge in the block-cut tree for each pair of a block and a cut vertex that belongs to that block. Then, $\partial(S)$ intersects at most one block of $G$. More precisely, for any two distinct blocks $B_1$ and $B_2$ of $G$, if $S \cap V(B_1) \neq \emptyset$ and $S \cap V(B_1) \neq V(B_1)$, then either $V(B_2) \subseteq S$, or $V(B_2) \subseteq V \setminus S$. Indeed, if this is not the case, then either $G[S]$ or $G[V \setminus S]$ would be disconnected. Thus, to solve \textsc{Largest st-Bond}, it is enough to consider, individually, each block on the path between $s$ and $t$ in the block-cut tree of $G$. Also, if a block is composed of a single edge, then it is a bridge in $G$, which is not a solution for the problem unless $k = 1$. Thus, we may assume without loss of generality that $G$ is 2-vertex-connected.

\textbf{Lemma 22.} Let $G$ be a 2-vertex-connected graph. For all $v \in V(G) \setminus \{s,t\}$, there is an $sv$-path and a $tv$-path which are internally disjoint.

\textbf{Proof.} Since $G$ is 2-vertex-connected, there are two disjoint $sv$-paths $P_s$ and $P'_s$ and there is a $tv$-path $P'_t$ which does not include $s$. Let $x$ be the first vertex of $P'_t$ which belongs to $V(P_s \cup P'_s)$ and assume, w.l.o.g., that $x \in P'_s$. Let $P''_s$ be the sub-path of $P'_s$ from $t$ to $x$ and $P''_t$ the sub-path of $P'_t$ from $x$ to $v$. Now define $P_s$ as $tP''_tXP''_sP'_sv$ and notice that $P_s$ is a tv-path disjoint from $P_t$.

\textbf{Lemma 23.} Let $G$ be a 2-vertex-connected graph. If $G$ contains $K_{2,2k}$ as a minor, then there exists $S \subseteq V(G)$ such that $\partial(S)$ is a bond of size at least $k$.

\textbf{Proof.} Let $G$ be a graph containing a $K_{2,2k}$ as a minor. If $k = 1$, the statement holds trivially, thus assume $k \geq 2$. Also, since $G$ is connected, one can assume that this minor was obtained by contracting or removing edges only, and thus its branch sets contain all vertices of $G$. Let $A$ and $B$ be the branch sets corresponding to first side of $K_{2,2k}$, and let $X_1, X_2, \ldots, X_{2k}$ be the remaining branch sets.

First, suppose that $s$ and $t$ are in distinct branch sets. If this is the case, then there exist distinct indices $a, b \in \{1, \ldots, 2k\}$ such that $s \in A \cup X_a$ and $t \in B \cup X_b$. Now observe that $G[A \cup X_a]$ and $G[B \cup X_b]$ are connected, which implies an st-bond with at least $2k - 1 \geq k$ edges. Now, suppose that $s$ and $t$ are in the same branch set. In this case, one can assume without loss of generality that $s, t \in A \cup X_{2k}$.

Define $U = A \cup X_{2k}$ and $Q = V(G) \setminus U$. Observe that $G[U]$ and $G[Q]$ are connected. Consider an arbitrary vertex $v$ in the set $Q$. Since $G$ is 2-vertex-connected, Lemma 22 implies that there exist an sv-path $P_s$ and a tv-path $P_t$ which are internally disjoint. Let $P'_s$ and $P'_t$ be maximal prefixes of $P_s$ and $P_t$, respectively, whose vertices are contained in $U$.

We partition the set $U$ into parts $U_s$ and $U_t$ such that $G[U_s]$ and $G[U_t]$ are connected. Since $G[U]$ is connected, there exists a tree $T$ spanning $U$. Direct all edges of $T$ towards $s$ and partition $U$ as follows. Every vertex in $P'_s$ belongs to $U_s$ and every vertex in $P'_t$ belongs to $U_t$. For a vertex $u \notin V(P'_s \cup P'_t)$, let $w$ be the first ancestor of $u$ (accordingly to $T$) which is in $P'_s \cup P'_t$. Notice that $w$ is well-defined since $u \in V(T)$ and the root of $T$ is $s \in V(P'_s \cup P'_t)$. Then $u$ belongs to $U_s$ if $w \in V(P'_s)$, and $u$ belongs to $U_t$ if $w \in V(P'_t)$.

Observe that that there are at least $2k - 1$ edges between $U$ and $Q$, and thus there are at least $k$ edges between $U_s$ and $Q$, or between $U_t$ and $Q$. Assume the former holds, as the other case is analogous. It follows that $G[U_s]$ and $G[U_t \cup Q]$ are connected and induce a bond of $G$ with at least $k$ edges.
Lemma \[\text{Lemma 21}\] and Lemma \[\text{Lemma 23}\] imply that there is an algorithm that either concludes that the input graph \(G\) has a bond of size at least \(k\), or outputs a tree-decomposition of an equivalent instance \(G'\) of width \(O(k)\).

\[\textbf{Corollary 24.}\] Given a graph \(G\), vertices \(s, t \in V(G)\), and an integer \(k\), there exists a polynomial-time algorithm that either concludes that \(G\) has an \(st\)-bond of size at least \(k\) or outputs a subgraph \(G'\) of \(G\) together with a tree decomposition of \(G'\) of width equals \(O(k)\), such that \(G'\) has an \(st\)-bond of size at least \(k\) if and only if \(G\) has an \(st\)-bond of size at least \(k\).

**Proof.** Find a block-cut tree of \(G\) in linear time \[\text{[11]}\], and let \(B_s\) and \(B_t\) be the blocks of \(G\) that contain \(s\) and \(t\), respectively. Remove each block that is not in the path from \(B_s\) to \(B_t\) in the block-cut tree of \(G\). Let \(G'\) be the remaining graph. For each block \(B\) of \(G'\), consider the vertices \(s'\) and \(t'\) of \(B\) which are nearest to \(s\) and \(t\), respectively. Using Lemmas \[\text{Lemma 21}\] and \[\text{Lemma 23}\] one can in polynomial time either conclude that \(B\) has an \(s't'\)-bond, in which case \(G\) is a yes-instance, or compute a tree decomposition of \(B\) with width at most \(O(k)\).

Now, construct a tree decomposition of \(G'\) as follows. Start with the union of the tree decompositions of all blocks of \(G'\). Next, create a bag \(\{u\}\) for each cut vertex \(u\) of \(G'\). Finally, for each cut vertex \(u\) and any bag corresponding to a block \(B\) connected through \(u\), add an edge between \(\{u\}\) and one bag of the tree decomposition of \(B\) containing \(u\). Note that this defines a tree decomposition of \(G'\) and that each bag has at most \(O(k)\) vertices. ◀

Note that since \(k\)-bonds are solutions for \textsc{Connected Max Cut}, the results presented in this section naturally apply to such a problem as well.

## 5 Taking the treewidth as parameter

In the following, given a tree decomposition \(T\), we denote by \(\ell\) one node of \(T\) and by \(X_\ell\) the vertices contained in the bag of \(\ell\). We assume w.l.o.g that \(T\) is a extended version of a nice tree decomposition (see \[\text{[9]}\]), that is, we assume that there is a special root node \(r\) such that \(X_r = \emptyset\) and all edges of the tree are directed towards \(r\) and each node \(\ell\) has one of the following five types: \textit{Leaf}; \textit{Introduce vertex}; \textit{Introduce edge}; \textit{Forget vertex}; and \textit{Join}.

Moreover, define \(G_\ell\) to be the subgraph of \(G\) which contains only vertices and edges that have been introduced in \(\ell\) or in a descendant of \(\ell\).

The number of partitions of a set of \(k\) elements is the \(k\)-th Bell number, which we denote by \(B(k)\) (\(B(k) \leq k!\) \[\text{[29]}\]).

\[\textbf{Theorem 25.}\] Given a nice tree decomposition of \(G\) with width \(tw\), one can find a bond of maximum size in time \(2^{O(tw \log tw)} \times n\) where \(n\) is the number of vertices of \(G\).

**Proof.** Let \(\partial_G(U)\) be a bond of \(G\), and \([U, V \setminus U]\) be the cut defined by such a bond. Set \(S_\ell^U = U \cap X_\ell\). The removal of \(\partial_G(U)\) partitions \(G_\ell[U]\) into a set \(C_\ell^U\) of connected components, and \(G_\ell[V \setminus U]\) into a set \(C_\ell^{V \setminus U}\) of connected components. Note that \(C_\ell^U\) and \(C_\ell^{V \setminus U}\) define partitions of \(S_\ell^U\) and \(X_\ell \setminus S_\ell^U\), denoted by \(\rho_1\) and \(\rho_2\) respectively, where the intersection of each connected component of \(C_\ell^U\) with \(S_\ell^U\) corresponds to one part of \(\rho_1\). The same holds for \(C_\ell^{V \setminus U}\) with respect to \(X_\ell \setminus S_\ell^U\) and \(\rho_2\).

We define a table for which an entry \(e[\ell, S, \rho_1, \rho_2]\) is the size of a largest cut-set (partial solution) of the subgraph \(G_\ell\), where \(S\) is the subset of \(X_\ell\) to the left part of the bond, \(X_\ell \setminus S\) is the subset to the right part, and \(\rho_1, \rho_2\) are the partitions of \(S\) and \(X_\ell \setminus S\) representing, after the removal of the partial solution, the intersection with the connected components to the left and to the right, respectively. If there is no such a partial solution then \(e[\ell, S, \rho_1, \rho_2] = -\infty\).
Computing the largest bond of a graph

For the case that $S$ is empty, two special cases may occur: either $U \cap V(G_{\ell}) = \emptyset$, in which case there are no connected components in $C_U^\ell$, and thus $\rho_1 = \emptyset$; or $C_U^\ell$ has only one connected component which does not intersect $X_{\ell}$, i.e., $\rho_1 = \{\emptyset\}$, this case means that the connected component in $C_U^\ell$ was completely forgotten. Analogously, we may have $\rho_2 = \emptyset$ and $\rho_2 = \{\emptyset\}$. Note that we do not need to consider the case $\emptyset \subsetneq \rho_i$ since it would imply a disconnected solution. The largest bond of a connected graph $G$ corresponds to the root entry $c[r, \emptyset, \{\emptyset\}, \{\emptyset\}]$.

To describe a dynamic programming algorithm, we only need to present the recurrence relation for each node type.

**Leaf:** In this case, $X_{\ell} = \emptyset$. There are a few combinations for $\rho_1$ and $\rho_2$: either $\rho_1 = \emptyset$, or $\rho_1 = \{\emptyset\}$, and either $\rho_2 = \emptyset$, or $\rho_2 = \{\emptyset\}$. Since for this case $G_{\ell}$ is empty, there can be no connected components, so having $\rho_1 = \emptyset$ and $\rho_2 = \emptyset$ is the only feasible choice.

$$c[\ell, S, \rho_1, \rho_2] = \begin{cases} 0 & \text{if } S = \emptyset, \rho_1 = \emptyset \text{ and } \rho_2 = \emptyset, \\ -\infty & \text{otherwise}. \end{cases}$$

**Introduce vertex:** We have only two possibilities in this case, either $v$ is an isolated vertex to the left ($v \in S$) or it is an isolated vertex to the right ($v \not\in S$). Thus, a partial solution on $\ell$ induces a partial solution on $\ell'$, excluding $v$ from its part.

$$c[\ell, S, \rho_1, \rho_2] = \begin{cases} c[\ell', S \setminus \{v\}, \rho_1 \setminus \{v\}, \rho_2] & \text{if } \{v\} \in \rho_1, \\ c[\ell', S, \rho_1, \rho_2 \setminus \{v\}] & \text{if } \{v\} \in \rho_2, \\ -\infty & \text{if } \{v\} \not\in \rho_1 \cup \rho_2. \end{cases}$$

**Introduce edge:** In this case, either the edge $\{u, v\}$ that is being inserted is incident with one vertex of each side, or the two endpoints are at the same side. In the former case, a solution on $\ell$ corresponds to a solution on $\ell'$ with the same partitions, but with value increased. In the latter case, edge $\{u, v\}$ may connect two connected components of a partial solution on $\ell'$.

$$c[\ell, S, \rho_1, \rho_2] = \begin{cases} c[\ell', S, \rho_1, \rho_2] + 1 & \text{if } u \in S \text{ and } v \not\in S \text{ or } u \not\in S \text{ and } v \in S, \\ \max_{\rho'_1} \{c[\ell', S, \rho'_1, \rho_2]\} & \text{if } u \in S \text{ and } v \in S, \\ \max_{\rho'_2} \{c[\ell', S, \rho_1, \rho'_2]\} & \text{if } u \not\in S \text{ and } v \not\in S. \end{cases}$$

Here, $\rho'_1$ spans over all refinements of $\rho_1$ such that the union of the parts containing $u$ and $v$ results in the partition $\rho_1$. The same holds for $\rho'_2$.

**Forget vertex:** In this case, either the forgotten vertex $v$ is in the left side of the partial solution induced on $\ell$, or is in the right side. Thus, $v$ must be in the connected component which contains some part of $\rho_1$, or some part of $\rho_2$. We select the possibility that maximizes the value

$$c[\ell, S, \rho_1, \rho_2] = \max_{\rho'_1, \rho'_2} \{c[\ell', S \cup \{v\}, \rho'_1, \rho_2], c[\ell', S, \rho_1, \rho'_2]\}. $$

Here, $\rho'_1$ spans over all partitions obtained from $\rho_1$ by adding $v$ in some part of $\rho_1$ (if $\rho_1 = \{\emptyset\}$ then $\rho'_1 = \{v\}$). The same holds for $\rho'_2$.

**Join:** This node represents the join of two subgraphs $G_{\ell'}$ and $G_{\ell''}$ and $X_{\ell'} = X_{\ell''} = X_{\ell'}$. By counting the bond edges contained in $G_{\ell'}$ and in $G_{\ell''}$, each edge is counted at least once, but edges in $X_{\ell}$ are counted twice. Thus

$$c[\ell, S, \rho_1, \rho_2] = \max\{c[\ell', S, \rho'_1, \rho'_2] + c[\ell', S, \rho''_1, \rho''_2]\} - |\{(u, v) \in E, u \in S, v \in X_{\ell} \setminus S\}|.$$
In this case, we must find the best combination between the two children. Namely, for \( i \in \{1, 2\} \), we consider combinations of \( \rho_i \) with \( \rho_i' \) which merge into \( \rho_i \). If \( \rho_i = \{\emptyset\} \) then either \( \rho_i' = \{\emptyset\} \) and \( \rho_i'' = \emptyset \); or \( \rho_i' = \emptyset \) and \( \rho_i'' = \{\emptyset\} \). Also, if \( \rho_i = \emptyset \) then \( \rho_i' = \emptyset \) and \( \rho_i'' = \emptyset \).

The running time of the dynamic programming algorithm can be estimated as follows. The number of nodes in the decomposition is \( O(tw \times n) \) [9]. For each node \( \ell \), the parameters \( \rho_1 \) and \( \rho_2 \) induce a partition of \( X_\ell \); the number of partitions of \( X_\ell \) is given by the corresponding Bell number, \( B(|X_\ell|) \leq B(tw + 1) \). Each such a partition \( \rho \) corresponds to a number of choice of parameter \( S \) that corresponds to a subset of the parts of \( \rho \); thus the number of choices for \( S \) is not larger than \( 2^{|\rho|} \leq 2^{|X_\ell|} \leq 2^{tw+1} \). Therefore, we conclude that the table size is at most \( O(B(tw+1) \times 2^{tw} \times tw \times n) \). Since each entry can be computed in \( 2^{O(tw \log tw)} \) time, the total complexity is \( 2^{O(tw \log tw)} \times n \). The correctness of the recursive formulas is straightforward.

The reason for the \( 2^{O(tw \log tw)} \) dependence on treewidth is because we enumerate all partitions of a bag to check connectivity. However, one can obtain single exponential-time dependence by modifying the presented algorithm using techniques based on Gauss elimination, as described in [9] Chapter 11 for Steiner Tree.

**Theorem 26.** Largest \( st \)-Bond is fixed-parameter tractable when parameterized by treewidth.

**Proof.** The solution of Largest \( st \)-Bond can be found by a dynamic programming as presented in Theorem 25 where we add \( s \) and \( t \) in all the nodes and we fix \( s \in S \) and \( t \notin S \).

Finally, the following holds.

**Corollary 27.** Largest Bond and Largest \( st \)-Bond are fixed-parameter tractable when parameterized by the size of the solution, \( k \).

**Proof.** Follows from Lemma 19, Lemma 21, Corollary 24, Theorem 25 and Theorem 26.

### 6 Infeasibility of polynomial kernels

As seen previously, any bond \( O(S) \) of a graph \( G \) intersects at most one of its block. Thus, an or-composition for Largest Bond parameterized by \( k \) can be done from the disjoint union of \( \ell \) inputs, by selecting exactly one vertex of each input graph and contracting them into a single vertex. Now, let \( (G_1, k, s_1, t_1), (G_2, k, s_2, t_2), \ldots, (G_\ell, k, s_\ell, t_\ell) \) be \( \ell \) instances of Largest \( st \)-Bond parameterized by \( k \). An or-composition for Largest \( st \)-Bond parameterized by \( k \) can be done from the disjoint union of \( G_1, G_2, \ldots, G_\ell \), by contracting \( t_i, s_{i+1} \) into a single vertex, \( 1 \leq i \leq \ell - 1 \), and setting \( s = s_1 \) and \( t = t_\ell \).

Therefore, the following holds.

**Theorem 28.** Largest Bond and Largest \( st \)-Bond do not admit polynomial kernel unless \( NP \subseteq coNP/poly \).

### References

1. Robert E. L. Aldred, Dries V. Dyck, Gunnar Brinkmann, Veerle Fack, and Brendan D McKay. Graph structural properties of non-Yutsis graphs allowing fast recognition. *Discrete Applied Mathematics*, 157(2):377–386, 2009.

2. Lawrence Christian Biedenharn and James D Louck. *The Racah-Wigner algebra in quantum theory*. Addison-Wesley, 1981.
Computing the largest bond of a graph

3. Hans L Bodlaender and Klaus Jansen. On the complexity of the maximum cut problem. *Nordic Journal of Computing*, 7(1):14–31, 2000.

4. Hans L Bodlaender, Jan Van Leeuwen, Richard Tan, and Dimitrios M Thilikos. On interval routing schemes and treewidth. *Information and Computation*, 139(1):92–109, 1997.

5. Jung Jin Cho, Yong Chen, and Yu Ding. On the (co)girth of a connected matroid. *Discrete Applied Mathematics*, 155(18):2456 – 2470, 2007.

6. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction To Algorithms*. MIT Press, 2001. URL: [https://books.google.co.in/books?id=NLngYyWFl_YC](https://books.google.co.in/books?id=NLngYyWFl_YC).

7. Bruno Courcelle, Johann A Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

8. Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

9. Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

10. Guoli Ding, Stan Dziobiak, and Haidong Wu. Large-or-minors in 3-connected graphs. *Journal of Graph Theory*, 82(2):207–217, 2016.

11. Dries V. Dyck and Veerle Fack. On the reduction of Yutsis graphs. *Discrete Mathematics*, 307(11):1506 – 1515, 2007. The Fourth Caracow Conference on Graph Theory.

12. Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April 1972.

13. Paul Erdős. On some extremal problems in graph theory. *Israel Journal of Mathematics*, 3(2):113–116, 1965.

14. Melissa Flynn. *The Largest Bond in 3-Connected Graphs*. PhD thesis, The University of Mississippi, 2017.

15. Fedor V Fomin, Petr A Golovach, Daniel Lokshtanov, and Saket Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM Journal on Computing*, 43(5):1541–1563, 2014.

16. Rajiv Gandhi, Mohammad T. Hajiaghayi, Guy Kortsarz, Manish Purohit, and Kanthi Sarpatwar. On maximum leaf trees and connections to connected maximum cut problems. *Information Processing Letters*, 129:31 – 34, 2018.

17. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

18. Michael R. Garey, David S. Johnson, and Robert E. Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

19. Christopher D. Godsil and Gordon F. Royle. *Algebraic Graph Theory*. Graduate texts in mathematics. Springer, 2001.

20. Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

21. Ralph E Gomory and Tien Chung Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

22. F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4(3):221–225, 1975.

23. D. J. Haglin and S. M. Venkatesan. Approximation and intractability results for the maximum cut problem and its variants. *IEEE Transactions on Computers*, 40(1):110–113, Jan 1991. [doi:10.1109/12.67327](https://doi.org/10.1109/12.67327)

24. Mohammad Taghi Hajiaghayi, Guy Kortsarz, Robert MacDavid, Manish Purohit, and Kanthi Sarpatwar. Approximation algorithms for connected maximum cut and related problems. In *In Proceedings of the 23rd European Symposium on Algorithms*, pages 693–704, 2015.
Petr Hliněný and Sang-il Oum. Finding branch-decompositions and rank-decompositions. *SIAM Journal on Computing*, 38(3):1012–1032, 2008.

S. Khot, G. Kindler, E. Mossel, and R. O’Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs. *SIAM Journal on Computing*, 37(1):319–357, 2007.

Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31(2):335–354, 1999.

Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

Andrew M Odlyzko. Asymptotic enumeration methods. *Handbook of combinatorics*, 2 (1063):1229, 1995.

James G. Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2006.

Christophe Picouleau. Complexity of the hamiltonian cycle in regular graph problem. *Theoretical Computer Science*, 131(2):463 – 473, 1994.

Saket Saurabh and Meirav Zehavi. Parameterized Complexity of Multi-Node Hubs. In *13th International Symposium on Parameterized and Exact Computation (IPEC 2018)*, pages 8:1–8:14, 2019.

Xiangkun Shen, Jon Lee, and Viswanath Nagarajan. Approximating graph-constrained max-cut. *Mathematical Programming*, 172(1):35–58, Nov 2018.

Dominic J.A. Welsh. Euler and bipartite matroids. *Journal of Combinatorial Theory*, 6(4):375 – 377, 1969.

A. P. Yutsis, V. V. Vanagas, and I. B. Levinson. *Mathematical apparatus of the theory of angular momentum*. Israel program for scientific translations, 1962.