Attempt on bounds for Primal Graphs

Aaryan Gupta

June 15 2022

1 What are Primal Graphs?

Let a CNF formula F have n variables $X = \{x_1, x_2, ..., x_n\}$ and k clauses $C = \{c_1, c_2, ..., c_k\}$. A primal graph $G = (V(G), E(G))$ corresponding to F is defined as $V(G) = X$ and with the edge set $E = \{(x_i, x_j) : \exists c \in C(x_i, x_j \in Var(c))\}$.

2 What is Primal Treewidth?

2.1 Tree Decomposition and Treewidth

For a Graph $G = (E(G), V(G))$, the tree $T = (V(T), E(T))$ together with a labelling function χ is defined to be the tree decomposition. χ is the labelling of vertices of T (referred to as nodes) by sets of vertices in G.

(T, χ) is a tree decomposition of G if and only if:

- For every $v \in V(G)$, $\exists\ t \in V(T)$ such that $v \in \chi(t)$
- For every $(v, w) \in E(G)$, $\exists\ t \in V(T)$ such that $v, w \in \chi(t)$
- For any three nodes $t_1, t_2, t_3 \in V(T)$, if t_2 lies on the unique path from t_1 to t_3, then $\chi(t_1) \cap \chi(t_3) \subseteq \chi(t_2)$

Now we define:
$\text{width}_{(T,\chi)} = \max_{t \in V(T)} |\chi(t)| - 1$
$\text{treewidth}_G = \min_{(T,\chi)} \text{width}_{(T,\chi)}$

2.2 Nice Tree Decomposition

For a graph G, any tree decomposition can be converted to a "nice" tree decomposition (T, χ, r) with a treewidth at most the treewidth of the former such that:

- Every node of T has at most two children
- If $t \in V(T)$ has two children t_1 and t_2, then $\chi(t) = \chi(t_1) = \chi(t_2)$ and t is called a *join node*.
- If $t \in V(T)$ has exactly one child t', then t is called a:
  a)*introduce node* such that $|\chi(t)| = |\chi(t')| + 1$ and $\chi(t') \subset \chi(t)$
  b)*forget node* such that $|\chi(t)| = |\chi(t')| - 1$ and $\chi(t) \subset \chi(t')$

Note that if $(T, \chi, r$ is nice, then $(T_t, \chi_{|V(T_t)}, t)$ is nice where T_t is the sub-tree of T rooted at node t.

2.3 Comment on implications of bounded primal treewidth on structure of CNF

As apparent from the name, primal treewidth is the treewidth of the primal graph of a CNF. We only look at nice tree decompositions with four types of nodes: join, introduce, forget, and leaf node. We want to analyse the impact of treewidth on $c_S(w)$ where S is the set of solutions of the CNF and $c_S(w)$ is the number of pairs of solutions that are at a hamming distance w from each other.

A low/bounded primal treewidth implies more tree-likeness in primal graph. This can intuitively mean:

- the clauses are small in size
- the clauses do not have too many variables in common

For a 2-CNF, a smaller treewidth would imply less common variables between the clauses. This would allow us to change a variable/literal from 1 to 0 with little or no consequence. This little or no consequence leads to a lower hamming distance between the two solutions. Intuitively, a low treewidth will imply solutions that are close by (low hamming distance) and a high treewidth will imply solutions that are far apart(high hamming distance). So for low vales of w, it makes sense that low treewidth will give higher $c_S(w)$ values and high treewidth will give lower $c_S(w)$ values. Analogously for higher values of w, low treewidth should give lower values of $c_S(w)$ and higher treewidth should give higher values of $c_S(w)$. For 2-CNFs we see that treewidth intuitively represents how far apart the solutions in the set S are.

We hope that we improve bounds for lower values of w as we vary w from 1 to $\lfloor log|S| \rfloor$ in the LICS paper. But given the relation, we see that a better parameter would be $(n - tw)$ instead of tw for lower w's. This is also the result we obtain.

3 Dynamic Programming Algorithm for bound on $c_S(w)$

This section was based on the approach for SAT using tree-decomposition. At each node in the tree decomposition, we maintain a table of values of $c_s(w)$ as well as n (like in #SAT). We try a bottom-up approach on calculating $c_s(w)$ starting at the leaf nodes and heading up and maintaining tables till the root. We describe what to do in this bottom-up traversal algorithm in the next sections.

3.1 Leaf Node

At the leaf node t, for each assignment $\alpha : \chi(t) \to \{0, 1\}$ we initialise as

$$c_S(w, t, \alpha) = \begin{cases} 0, & \text{if } w \geq 1 \text{ or if } \alpha \text{ falsifies some } C \in F \\ 1, & \text{otherwise} \end{cases}$$

Note that we can initialise it to 1 without conditions if we want an upper bound and this step takes time.

3.2 Introduce Node

Let the introduce node t have a child t'. For each assignment $\alpha : V_t \to \{0, 1\}$ we do

$$c_S(w, t, \alpha) = \begin{cases} 0, & \text{if } \alpha \text{ falsifies some } C \in F \\ c_S(w, t', \alpha_{|\chi(t')}), & \text{otherwise} \end{cases}$$

To bound $c_S(w)$ we can ignore the case when we set it to 0 and can only consider the second value for faster results.

3.3 Join Node

Let t be the join node with children t_1 and t_2. We compute $c_S(w, t, \alpha)$ as:

$$c_S(w, t, \alpha) = \sum_{i=0}^{w} c_S(i, t_1, \alpha).c_S(w - i, t_2, \alpha)$$

Proof Outline. We know that $(V_{t_1} \backslash \chi(t)) \cup (V_{t_2} \backslash \chi(t)) = \phi$ because of the connectivity rule for nice tree decompositions. Let us divide the variables involved into three disjoint sets $A = (V_{t_1} \backslash \chi(t)$, $B = (V_{t_2} \backslash \chi(t)$, and $C = \chi(t)$. The set of assignments for node t have all three sets A, B, C involved while node t_1 has A, C involved and t_2 has B, C involved. The part of the assignments for A are the same so there is no hamming distance there. So we partition the hamming distance between sets of variables B and C as i and $w - i$ and take the convolution sum.

3.4 Forget Node

Let the forget node be t with a child t' such that $\chi(t') = \chi(t) \cup \{x\}$, where x is a single variable For this node, we have not been able to find exact relations at all due to the difficulty of not knowing the hamming distance between the cross terms. Here I present some upper bounds that we have come up with.

The First Bound

$$c_S(w, t, \alpha) \leq c_S(w, t', \alpha \cup \{x, 0\}) + c_S(w, t, \alpha \cup \{x, 1\}) + c_S(0, t', \alpha \cup \{x, 0\}).c_S(0, t', \alpha \cup \{x, 1\})$$

Proof Outline. For each assignment τ in node t, we have two possible assignments $\tau \cup \{x, 0\}$ and $\tau \cup \{x, 1\}$ in node t'. So, if we look at a pair of assignments (τ_1, τ_2) in node t with a hamming distance of w, we can have four possible pairs in node t' which are $(\tau_1 \cup \{x, 0\}, \tau_2 \cup \{x, 1\})$, $(\tau_1 \cup \{x, 1\}, \tau_2 \cup \{x, 0\})$, $(\tau_1 \cup \{x, 0\}, \tau_2 \cup \{x, 0\})$, $(\tau_1 \cup \{x, 1\}, \tau_2 \cup \{x, 1\})$. The first two pairs would be included in $c_S(w, t', \alpha \cup \{x, 0\})$ and $c_S(w, t', \alpha \cup \{x, 1\})$ respectively. The cross terms are upper bounded by $c_S(0, t', \alpha \cup \{x, 0\}).c_S(0, t', \alpha \cup \{x, 1\})$ as $w = 0$ basically represents the number of solutions and this products gives the total number of pairs. This is a huge over-approximation. we will try to bring this down in the next couple of tries.

Here, I have documented some thoughts Yash and me had about cross terms which might lead us to a better bound.

Thoughts for $w = 1$: We know that the cross terms have at least one different value, which is at x. So, the rest of the strings excluding x need to be the same. This leads us to an obvious bound-

$$c_S(1, t, \alpha) \leq c_S(1, t', \alpha \cup \{x, 0\}) + c_S(1, t, \alpha \cup \{x, 1\}) + \min\{c_S(0, t', \alpha \cup \{x, 0\}), c_S(0, t', \alpha \cup \{x, 1\})\}$$

Thoughts for $w = 2$: Here, the two strings excluding x differ at exactly one different position. This leads to the upper-bound-

$$c_S(2, t, \alpha) \leq c_S(2, t', \alpha \cup \{x, 0\}) + c_S(2, t, \alpha \cup \{x, 1\}) + \min\{c_S(0, t', \alpha \cup \{x, 0\}), c_S(0, t', \alpha \cup \{x, 1\})\}.|\text{Var}(V_{t'} \setminus \chi(t'))|$$

Thoughts for $w = 3$: Here the two strings excluding x differ at exactly two positions. One obvious bound would be the combinatorial approach as above-

The Second Bound

Proof Outline.

3.5 Root Node

4 Dynamic Programming algorithm for bound on $\sum_w c_S(w)$ or $e_{\leq w}(S)$