Invited Paper: Approximation Algorithms via Lattice-Linear Predicates

Robert P. Streit and Vijay K. Garg

The University of Texas at Austin, Austin, TX 78712, USA
{rpstreit,garg}@utexas.edu

Abstract. We show that the maximal-matching 2-approximation for vertex cover and the classical primal-dual f-approximation for weighted set cover can be formulated as lattice-linear predicate (LLP) computations on canonical sublattices induced by deterministic lexicographic rules. In each case, the forbidden predicate identifies pairwise nonconflicting coordinates that may advance simultaneously, making the algorithm's parallel structure explicit. We then study weighted set cover subject to definite Horn constraints. For binary implications, corresponding to width-two clauses, we derive an LLP primal-dual f-approximation by projecting the dual onto closure inequalities and computing advances through parametric min-cuts. In contrast, for width-three clauses, no polynomial-time constant-factor approximation exists unless $\mathsf{P} = \mathsf{NP}$.

1 Introduction

A *vertex cover* of an undirected graph (V, E) is a set $C \subseteq V$ meeting every edge. A *set cover* of a universe $\mathcal{U}$ is a subfamily of a given collection whose union is $\mathcal{U}$. Vertex cover is the special case in which each element (an edge) lies in exactly two sets (its two endpoints). Both problems are NP-hard and admit classical approximation algorithms [35]. Specifically, a maximal matching yields a vertex cover of size at most $2 \cdot \mathrm{OPT}$. In the weighted setting, the primal-dual method of Bar-Yehuda and Even [3] maintains this approximation and, more generally, gives an f-approximation for weighted set cover where f is the maximum number of sets sharing a common element.

In this work, we describe these mechanisms as computing the least fixpoint of an appropriate function in a distributive lattice. We show these problems, and certain constrained variants, can be solved via the *lattice-linear predicate* (LLP) Algorithm [11]. In the LLP model the state is a vector G over an index set ordered componentwise. A *forbidden* predicate marks at every state the coordinates that must increase, and an *advance* rule increases them. When the predicate is lattice-linear its satisfying states are closed under meet, so a unique least fixpoint exists and is reached by advancing forbidden coordinates from the bottom state in *any* order, and in particular in parallel, since the forbidden coordinates of a round remain forbidden until they themselves are advanced.

For this, the LLP algorithm has given a conceptually simple view of parallel algorithm design for a number of problems we overview in the related work, which

recent experiments have shown yields competitive runtimes compared with state-of-the-art parallel combinatorial optimization algorithms [2]. Furthermore, the LLP algorithm is naturally asynchronous because advancements can be made in any order (even on stale information). This has led to applications in self-stabilizing and asynchronous distributed computing algorithms [17, 19, 20, 18].

The difficulty is that classical approximation algorithms do not fit the LLP model as readily as exact ones do. Previous LLP algorithms compute exact optima: stable marriages, shortest paths, and the assignment problem [11], minimum spanning trees [1], min-cuts [34] and others. For those problems the exact solutions are naturally meet-closed, either by way of uniqueness or well-known lattice representation theorems [32, 21]. For an approximation algorithm the natural property is feasibility and a certification of approximate value. This is a looser structure, and in general gives many minimal incomparable approximate feasible solutions under some natural order (which cannot be meet-closed).

Our fix is to use *deterministic lexicographic tie-breaking* to define a suitable sublattice on the solution space confining the resulting LLP computation. Specifically, for each of our algorithms we show the possible states are always lesser than that of a solution computed by a greedy algorithm with lexicographic tie-breaking, and that on this *principal ideal* (that is, the sublattice of all states less than or equal to that of the greedy solution) the computation primitives fit the LLP model in a way recovering classical approximation guarantees. Our work thus extends the LLP framework from exact optimization to approximation.

In general, formulating algorithms in the LLP framework pays off in three ways. First, *uniformity*: all LLP algorithms reduce to the same three-step recipe — choose a lattice, exhibit a lattice-linear forbidden predicate, and give the advance function. Second, *explicit parallelism*: the forbidden predicate exposes, at each state, a maximal pairwise non-conflicting batch of coordinates that advance together in one step, so every algorithm is inherently parallel and its depth is the number of rounds. Third, the framework *generally composes*: specifically, conjunctions of lattice-linear predicates remain lattice-linear, which has led to LLP algorithms for constrained optimization problems [34, 15]. However, this phenomenon is more nuanced for approximation algorithms (since approximate solutions are not easily described as lattice-linear predicates). In particular, while constraints defined by Horn clauses have been successfully composed with LLP algorithms for exact problems [34], Section 5 shows a sharp contrast for approximation: constraining by width-two Horn clauses (i.e. binary implications) still yields f-approximation LLP algorithms for weighted set cover, whereas width-three clauses make any constant-factor approximation NP-hard.

Contributions. In summary, this paper makes the following contributions:

- An LLP formulation of 2-approximate vertex covers via maximal matching. In each round the forbidden predicate selects a pairwise non-conflicting batch of coordinates, so the algorithm is explicitly parallel (Section 3).
- An LLP formulation of the primal-dual f-approximation for weighted set cover, due to Bar-Yehuda and Even [3]. The forbidden predicate raises slack

objects which are non-interacting (with lexicographic tie-breaking); this makes the predicate lattice-linear and parallel updates conflict-free (Section 4).
– An investigation into solving covering problems under constraints defined by Horn clauses using LLP algorithms. We extend the primal-dual weighted set cover algorithm to the setting with width-two clauses using network flow and min-cut techniques. The forbidden predicate now raises slack objects belonging to different components in the implication graph for conflict-free parallel updates (Section 5.1). Then, we establish a hardness boundary by showing any constant-factor approximation algorithm to the setting with width-three Horn constraints can be used to solve 3-SAT (Section 5.2).

Related work. The greedy set-cover heuristic and its H_n analysis are due to Johnson [24], Lovász [28], and Chvátal [5]; Hochbaum [22] gave the LP-based vertex-cover approximation, and Bar-Yehuda and Even [3, 4] the local-ratio and primal-dual method. Vazirani [35] and Williamson and Shmoys [36] are standard references for approximation algorithms, and Goemans and Williamson [16] survey the primal-dual method. On the hardness side, Dinur and Safra [7] show that vertex cover is NP-hard to approximate within a factor of 1.36, and Khot and Regev [25] show that it is hard to approximate within $2 - \varepsilon$ under the Unique Games Conjecture. The LLP framework [11] builds on lattice-theoretic methods for combinatorial optimization [10]; it has been applied to dynamic programming [13], housing allocation [12], minimum spanning trees [1], min-cuts [34], multiplication and modulo [20] and generalizations of stable matching [15]. Gupta and Kulkarni also extend lattice-linearity to self-stabilizing algorithms [17, 19, 18, 20]. In [18] these authors also examine problems which are not easily cast in the LLP framework. However, while their method stabilizes into a region on which the predicate becomes lattice-linear, our work instead identifies sublattices containing every reachable state and proves lattice-linearity on that structure. A forthcoming book gives a uniform treatment [14].

For parallel approximation algorithms, Luby [29] and Israeli and Itai [23] give parallel maximal matching (hence a parallel 2-approximate vertex cover); Khuller, Vishkin, and Young [26] give a parallel primal-dual approximation for weighted vertex and set cover; and Rajagopalan and Vazirani [33] give RNC primal-dual approximations for set cover. These algorithms are problem-specific. The LLP formulation instead obtains parallel algorithms from a single framework. Finally, [30] give a primal-dual algorithm for a class of problems related to that studied in Section 5.1. Their formulation modifies the primal program, whereas ours projects the dual onto closure inequalities and uses network flow and min-cut machinery. As a result, we achieve incomparable approximation ratios. Furthermore, our method is explicitly parallel while that of [30] is not.

2 Preliminaries

We work with the Lattice-Linear Predicate (LLP) framework of [11]. A *lattice* is a poset $(\mathcal{L}, \leq)$ in which every pair x, y has a meet $x \sqcap y$ and a join $x \sqcup y$, with

a bottom $\bot$ and top $\top$. The algorithms explore a distributive lattice, which will always be a lattice of vectors indexed by a set I ordered componentwise. We call $G \in \mathcal{L}$ a *state* and $G[i]$ its component at $i \in I$. A *predicate* is a map $B : L \to \{true, false\}$; an LLP computation seeks the least state G with $B(G) = true$.

Definition 1 (Forbidden index). *Given a state G with $\neg B(G)$, an index $i \in I$ is* forbidden *in G if every $H \geq G$ with $H[i] = G[i]$ also satisfies $\neg B(H)$.*

In English, i is forbidden whenever the predicate cannot be met without increasing (i.e. *advancing*) $G[i]$. This leads to *lattice-linearity*.

Definition 2 (Lattice-linear predicate). *A predicate B is* lattice-linear *on $\mathcal{L}$ if for every state G with $\neg B(G)$ at least one index is forbidden in G, and such an index can be found efficiently.*

We remark that the existence of a forbidden index in every state G with $\neg B(G)$ is equivalent to the satisfying elements of the predicate being meet-closed [11]. So, our lemmas showing lattice-linearity on chosen sublattices also show meet-closure on the sublattices. Lattice-linearity gives a universal algorithm.

Theorem 1 ([11]). *If B is lattice-linear, then $\{G \in L : B(G)\}$ is closed under meet and hence has a unique least element. This least element is computed by starting at $\bot$ and repeatedly advancing forbidden indices; because a forbidden index stays forbidden until it is itself advanced, the forbidden indices of a round may be advanced in parallel, and the computation reaches the least fixpoint regardless of the order of advancements.*

For a finite lattice, termination follows from finite height. For the continuous price lattices of Sections 4 and 5, this follows instead from the advance rule: each advance makes a new object tight, and only finitely many such events can occur.

The design of any LLP algorithm amounts to (i) choosing the lattice, (ii) exhibiting a lattice-linear forbidden predicate, and (iii) giving the advance function. A difficulty is that our predicates will generally not be lattice-linear on the full lattice of solutions. However, by imposing lexicographic ordering conditions in the forbidden predicates, we will see the predicates are lattice-linear on sublattices containing the entire space of possible intermediate states computed by our algorithms. This allows us to correctly apply LLP methods.

3 Vertex Cover via a Lex-Minimal Maximal Matching

Given a graph (V, E), it is classically known that a maximal matching induces a 2-approximation for vertex cover by selecting the endpoints of the edges in the matching [35]. We now give an LLP algorithm based upon this observation.

Throughout this section, fix a lexicographic order $<_{\text{lex}}$ over E using the endpoints. We work in the Boolean lattice $\{0, 1\}^E$ ordered componentwise. The state G records the currently selected edges, so $G[e] = 1$ if the edge e is currently

Algorithm LLP-LexMatchVertexCover: Parallel 2-approximation for vertex cover via lex-minimal maximal matching.

Input: Graph (V, E) with vertices labeled $1, \ldots, n$
Output: $C \subseteq V$: a vertex cover

G: array[E] of boolean, initially $\forall e \in E :\ G[e] := 0$ `// matching edges`
forbidden (e) : $minuncovered_G(e)$
$\Rightarrow$ **advance** : $G[e] := 1$
return $C := \{v \in V : \exists\, e \in E \text{ with } G[e] = 1 \wedge v \in e\}$

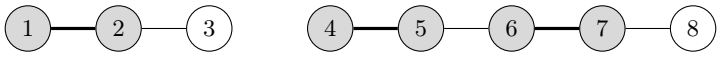

Fig. 1. The instance of Example 1: paths $1-2-3$ and $4-5-6-7-8$. LLP-LexMatchVertexCover selects the lex-minimal edges $(1,2)$ and $(4,5)$ (bold) in the first parallel round and $(6,7)$ in the second; their endpoints (shaded) form the cover $\{1,2,4,5,6,7\}$, which is exactly 2 OPT since the optimum is $\{2,5,7\}$.

selected and 0 otherwise. An edge e is *uncovered* at G if no selected edge shares an endpoint with it,

$$uncovered_G(e) \ \equiv\ \forall f \in E :\ G[f] = 1 \implies e \cap f = \emptyset,$$

and *min-uncovered* when it is lex-minimal among adjacent uncovered edges,

$$minuncovered_G(e) \equiv uncovered_G(e), \\ \wedge\ \forall f \in E : (uncovered_G(f) \wedge e \cap f \neq \emptyset) \implies e \leq_{\text{lex}} f.$$

We use $forbidden_G \equiv minuncovered_G$, so our LLP algorithm computes,

$$B_{\text{vc}}(G) \ \equiv\ \forall e \in E :\ \neg\, minuncovered_G(e),$$

through advancements defined by the update $G[e] := \mathit{true}$ for forbidden e. The resulting method is shown in LLP-LexMatchVertexCover. The method computes a maximal matching and returns a vertex cover C by taking the endpoints. We illustrate the resulting computation on a small instance before identifying the lattice on which B_{vc} is lattice-linear and verifying correctness.

Example 1. Let $V = \{1, \ldots, 8\}$ and $E = \{(1,2), (2,3), (4,5), (5,6), (6,7), (7,8)\}$, see Figure 1, ordered $(1,2) <_{\text{lex}} (2,3) <_{\text{lex}} (4,5) <_{\text{lex}} (5,6) <_{\text{lex}} (6,7) <_{\text{lex}} (7,8)$. Write $e_1 = (1,2)$, $e_2 = (2,3)$, $e_3 = (4,5)$, $e_4 = (5,6)$, $e_5 = (6,7)$, $e_6 = (7,8)$. The run takes two parallel rounds:
Round 1. Every edge is uncovered. Each of e_1 and e_3 is lex-smaller than its uncovered neighbour (e_2 and e_4, respectively), and the two are non-adjacent, so both are lex-minimal and advance in parallel. Selecting them covers $1, 2, 4, 5$, so e_2 and e_4 become covered, leaving e_5 and e_6 uncovered.
Round 2. Among the remaining uncovered edges, $e_5 = (6,7)$ is lex-smaller than its uncovered neighbour $e_6 = (7,8)$, so e_5 is lex-minimal and advances while e_6 waits; selecting e_5 covers $6, 7$ and hence e_6.

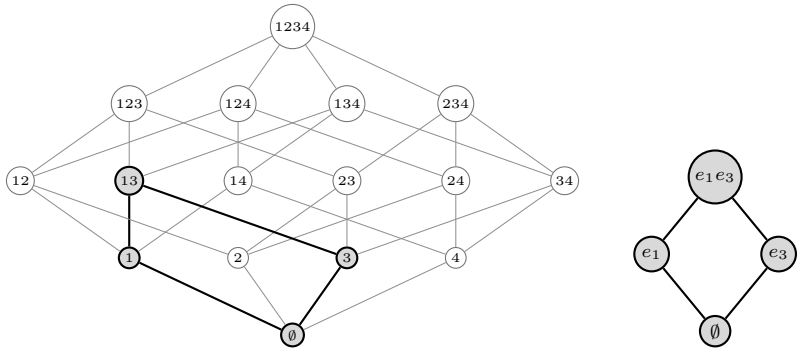

Fig. 2. The two lattices for a smaller four-edge instance (the paths 1–2–3 and 4–5–6), with edges labelled $1 = e_1 = (1,2)$, $2 = e_2 = (2,3)$, $3 = e_3 = (4,5)$, $4 = e_4 = (5,6)$. Left: the full Boolean lattice $\{0,1\}^E$ (16 states; a subset is written as its list of edge indices). The shaded nodes and bolded edges are the sublattice $\mathcal{L}_{\text{vc}}$. Right: $\mathcal{L}_{\text{vc}}$ on its own, a four-element distributive lattice whose top $\{e_1, e_3\}$ is the unique state satisfying B_{vc} (which is the fixpoint the algorithm reaches).

Termination. No edge is uncovered. The output is $C = \{1,2,4,5,6,7\}$ of size 6, while the optimum is $\{2,5,7\}$ of size 3.

For the path $e_1 = (1,2) <_{\text{lex}} e_2 = (2,3)$ both $\{e_1\}$ and $\{e_2\}$ leave no edge uncovered, yet their meet $\varnothing$ does. So on the full lattice $\{0,1\}^E$ we find B_{vc} is not meet-closed, and so is not lattice-linear. But, the algorithm can never visit a state like $\{e_2\}$. Specifically, it selects only *lex-minimal* edges, and such edges are contained in a *sublattice* of $\{0,1\}^E$ on which B_{vc} *is* lattice-linear. The correct LLP interpretation is then obtained by restricting to this lattice: let m_1 be the lex-minimal edge of (V,E). Delete m_1 and every edge adjacent to it, let m_2 be the lex-minimal edge of what remains, and repeat until no edge is left. The union $M^* = \bigcup_i \{m_i\}$ is the lex-greedy maximal matching, and its edges are exactly the *selectable* ones. Every state reachable by the algorithm has support contained in M^*, so define,

$$\mathcal{L}_{\text{vc}} = \left\{ G \in \{0,1\}^E : G[e] = 0 \text{ for every } e \notin M^* \right\},$$

which is a sublattice of $\{0,1\}^E$. Figure 2 draws both lattices for a smaller four-edge instance. Lattice-linearity on $\mathcal{L}_{\text{vc}}$ now follows.

Lemma 1. *Select $G \in \mathcal{L}_{\text{vc}}$. If $\neg B_{\text{vc}}(G)$, then all of the following are true:*

(i) *some edge e is min-uncovered,*
(ii) *advancing $G[e] := 1$ remains in $\mathcal{L}_{\text{vc}}$,*
(iii) *$\neg B_{\text{vc}}(H)$ holds for all $H \in \mathcal{L}_{\text{vc}}$ with $H \geq G$ and $H[e] = G[e]$.*

Proof. Suppose $\neg B_{\text{vc}}(G)$. Then some edge e is min-uncovered by the definition of B_{vc}. This gives (i). For (ii), by way of contradiction suppose $e \notin M^*$. By the

construction of M^*, the edge e was deleted precisely when some edge $m \in M^*$ adjacent to e was chosen. Since m was chosen and not e, this makes $m <_{\text{lex}} e$. So, for e to be min-uncovered in G it follows that m is covered in G. But the support of G is a subset of M^*, so m is covered only if $G[m] = 1$. This implies e is covered, a contradiction. This shows $e \in M^*$, i.e. (ii). Finally, select $H \in \mathcal{L}_{\text{vc}}$ with $H[e] = G[e]$. We've shown $e \in M^*$, and so e must be uncovered in H as the support of H is a subset of the matching M^*. The set of uncovered edges is decreasing as the state advances, so $H \geq G$ implies e remains lex-minimal among uncovered edges in H. So e is min-uncovered in H, hence (iii). □

Theorem 2. *LLP-LexMatchVertexCover returns a vertex cover C satisfying $|C| \leq 2\text{OPT}$.*

Proof. Lemma 1 shows B_{vc} is lattice-linear on $\mathcal{L}_{\text{vc}}$, so forbidden indices correctly remain forbidden until they are advanced. Hence, a min-uncovered edge remains min-uncovered until it is advanced, meaning the support of the final state G will be a matching. At termination, every edge is covered, so has an endpoint incident to some edge in the matching. So $C = \bigcup_{e:G[e]=1} e$ is a vertex cover.

The rest is classic: every vertex cover must contain at least one endpoint of each edge in the matching defined by G, so $\text{OPT} \geq \sum_{e \in E} G[e]$. Since $|C| = 2\sum_{e \in E} G[e]$, this implies $|C| \leq 2\text{OPT}$ as desired. □

4 Weighted Set Cover via Primal-Dual Methods

With vertex weights $w : V \to \mathbb{R}_{\geq 0}$ the matching argument fails (an edge (a, b) with $w(a) = 1$, $w(b) = K$ gives ratio $1 + K$). The primal-dual method instead maintains a price $G[e] \geq 0$ on each edge and raises prices until every edge is paid for by a tight endpoint. Note that weighted vertex cover is just a weighted set cover instance, where the elements are the edges and the sets are the edge-incidence sets corresponding to each vertex. The primal-dual method of [3] extends to any weighted set cover instance to obtain an f-approximation, where f is the maximum frequency of any element (note that in a transformed vertex cover instance $f = 2$, since each edge is incident to two vertices). We will give an LLP algorithm based upon this pricing mechanism.

Let $\mathcal{U}$ be a universe on n elements and $\mathcal{S}$ a collection of m sets $\{Y_1, \ldots, Y_m\}$. Impose a lexicographic order $\leq_{\text{lex}}$ over $\mathcal{U}$, and assume that the instance is feasible (which is for all $x \in \mathcal{U}$ there exists $Y \in \mathcal{S}$ with $x \in Y$). The state is now a *price* $G[x] \geq 0$ for each $x \in \mathcal{U}$, raised until every element lies in a tight set. These notions come from the linear programming relaxation, where the prices are the dual variables and tight sets correspond to tight dual constraints (see Section 5), though the present method is explainable without linear programming techniques. Finally, note that by assuming the instance is feasible, raising an element's price until a containing set becomes tight is always well defined.

Working in the continuous lattice $\mathbb{R}^{\mathcal{U}}_{\geq 0}$ of element prices, write $\mathit{price}_G(Y) = \sum_{x \in Y} G[x]$. A set is *tight* when $\mathit{price}_G(Y) = w(Y)$, i.e. its cumulative price

Algorithm LLP-WeightedSetCover: A parallel f-approximation for weighted set cover via lex-minimal element pricing.

Input: Universe $\mathcal{U}$; collection $\mathcal{S}$; weights $w : \mathcal{S} \to \mathbb{R}_{\geq 0}$
Output: $\mathcal{C} \subseteq \mathcal{S}$: a set cover

G: array[U] of real, initially $\forall x \in \mathcal{U} :\ G[x] := 0$ `// element prices`
forbidden (x) : $\mathit{slackmin}_G(x)$
$\Rightarrow$ **advance** : $G[x] := G[x] + \min\{w(Y) - \mathit{price}_G(Y) : Y \in \mathcal{S}, x \in Y\}$
return $\mathcal{C} := \{Y \in \mathcal{S} : \mathit{price}_G(Y) = w(Y)\}$

meets its weight, and an element is *slack* when it lies in no tight set:

$$\mathit{slack}_G(x) \equiv \neg\exists Y \in \mathcal{S} : x \in Y \wedge \mathit{price}_G(Y) = w(Y).$$

Call an element *slack-minimal* if it is slack and lex-minimal among other slack elements contained in a common set,

$$\mathit{slackmin}_G(x) \equiv \mathit{slack}_G(x), \\ \wedge\ \forall z \in \mathcal{U} : \mathit{slack}_G(z) \wedge (\exists Y \in \mathcal{S} : \{z, x\} \subseteq Y) \implies x \leq_{\text{lex}} z.$$

We will use the forbidden predicate $\mathit{forbidden}_G \equiv \mathit{slackmin}_G$, and advancing x will raise $G[x]$ until some set containing it becomes tight. Thus, the advance is $G[x] := G[x] + \min_{Y \in \mathcal{S} : x \in Y} w(Y) - \mathit{price}_G(Y)$. The *forbidden* predicate advances a slack element only when it is lex-minimal among the slack elements sharing a set with it. Distinct lex-minimal slack elements share no set, so raising them in parallel charges each set's budget through at most one element. This maintains feasibility, and will assist in Lemma 2 in showing lattice-linearity. The predicate the algorithm computes is,

$$B_{\text{wsc}}(G) \equiv \forall x \in \mathcal{U} :\ \neg \mathit{slackmin}_G(x).$$

The algorithm is shown in LLP-WeightedSetCover. The method returns the tight sets at termination. Observe that an element is never slack after it advances, so the algorithm terminates in at most n advancements.

As in the last section, B_{wsc} is generally not lattice-linear on all of $\mathbb{R}^{\mathcal{U}}_{\geq 0}$. To identify a suitable lattice, let G^* be the price vector given by (sequentially) increasing slack elements with lexicographic tie-breaking. This is, beginning with the bottom element (i.e. $\mathbf{0}$), select the lex-minimal slack element z_1 and increase its price until a set containing z_1 becomes tight (equivalently until z_1 is no longer slack). Repeat with z_2 and so on until no slack element exists to obtain G^*. It is important to note that G^* is feasible (in the sense that $\mathit{price}_G(Y) \leq w(Y)$ for all $Y \in \mathcal{S}$) and no element is slack at the end of this process. Define,

$$\mathcal{L}_{\text{wsc}} \triangleq \{G \in \mathbb{R}^{\mathcal{U}}_{\geq 0} : G \leq G^*\},$$

i.e. $\mathcal{L}_{\text{wsc}}$ is the principal ideal generated by G^*. We show that the advancements made by LLP-WeightedSetCover can never attain a state beyond $\mathcal{L}_{\text{wsc}}$, and that B_{wsc} is lattice-linear on $\mathcal{L}_{\text{wsc}}$.

Lemma 2. *Select $G \in \mathcal{L}_{\text{wsc}}$. If $\neg B_{\text{wsc}}(G)$, then all of the following are true:*

(i) *some element $x \in \mathcal{U}$ is slack-minimal,*
(ii) *advancing $G[x] := G[x] + \min_{Y \in \mathcal{S}: x \in Y} w(Y) - price_G(Y)$ remains in $\mathcal{L}_{\text{wsc}}$,*
(iii) *for all $H \in \mathcal{L}_{wsc}$, if $H \geq G$ and $H[x] = G[x]$ then $\neg B_{\text{wsc}}(H)$.*

Proof. The first item (i) follows by definition of B_{wsc}. Fix such slack-minimal $x \in \mathcal{U}$. For (ii), by way of contradiction suppose that,

$$G^*[x] < G[x] + \min_{Y \in \mathcal{S}: x \in Y} w(Y) - price_G(Y)$$

This implies x is slack in the state $\widehat{G}$ obtained by the advancement $G[x] := G^*[x]$. So for x to not be slack in G^*, there must exist $z_i \neq x$ contained in a set containing x with $G^*[z_i] > \widehat{G}[z_i]$. Let i be the first index satisfying this among elements selected in the construction of G^*. Then z_j is such that there exists no $Y \in \mathcal{S}$ with $\{x, z_j\} \subseteq Y$ or $G^*[z_j] \leq \widehat{G}[z_j]$ for all $j < i$. Therefore, the price of every set containing x according to $\widehat{G}$ is greater than or equal to that in the construction of G^* at the time z_i is selected. So x is slack when z_i is selected, meaning that $z_i <_{\text{lex}} x$ must hold. Therefore, for x to be slack-minimal in G, z_i cannot be slack in G. Select a set $Y \in \mathcal{S}$ with $Y \ni z_i$ and $price_G(Y) = w(Y)$. But $G \leq G^*$ and $G^*[z_i] > G[z_i]$ implies $price_G(Y) < price_{G^*}(Y)$. This makes $price_{G^*}(Y) > w(Y)$ which contradicts the construction of G^*. And so (ii) follows.

Finally, for (iii) select $H \in \mathcal{L}_{\text{wsc}}$ with $H \geq G$ and $H[x] = G[x]$. Observe that the set of slack elements is decreasing with increasing state. So, as long as x is slack in H then it is slack-minimal in H. Well, (ii) shows $H[x] < G^*[x]$, and so by $H \leq G^*$,

$$price_H(Y) < price_{G^*}(Y), \quad \forall Y \in \mathcal{S} : x \in Y.$$

In particular, no set containing x can be tight since G^* is feasible, $price_{G^*}(Y) \leq w(Y)$ for all $Y \in \mathcal{S}$. So x is slack in H, and $\neg B_{wsc}(H)$ follows. □

Theorem 3. *Let $f := \max_{x \in \mathcal{U}} |\{Y \in \mathcal{S} : x \in Y\}|$ be the maximum element frequency. LLP-WeightedSetCover returns a set cover $\mathcal{C}$ with $\sum_{Y \in \mathcal{C}} w(Y) \leq f \cdot$ OPT.*

Proof. Let G be the terminal state. Observe that $B_{\text{wsc}}(G)$ holds as $\mathcal{L}_{\text{wsc}}$ contains a state satisfying the predicate (i.e. G^*) so no element is slack in G. For any choice of set cover $\mathcal{C}^* \subseteq \mathcal{S}$ this implies,

$$\sum_{x \in \mathcal{U}} G[x] \leq \sum_{x \in \mathcal{U}} G[x] \cdot |\{Y \in \mathcal{C}^* : x \in Y\}| = \sum_{Y \in \mathcal{C}^*} \sum_{x \in Y} G[x] \leq \sum_{Y \in \mathcal{C}^*} w(Y). \quad (1)$$

Therefore, $\sum_{x \in \mathcal{U}} G[x] \leq$ OPT. But because every set in $\mathcal{C}$ is tight, we also have,

$$\sum_{Y \in \mathcal{C}} w(Y) = \sum_{x \in \mathcal{U}} G[x] \cdot |\{Y \in \mathcal{C} : x \in Y\}| \leq f \sum_{x \in \mathcal{U}} G[x] \leq f \cdot \text{OPT}.$$

This proves the theorem. □

5 Weighted Set Cover with Definite Horn Constraints

A *Horn clause* is a disjunction with at most one positive literal, and is *definite* if it contains exactly one positive literal. Associate with each $Y \in \mathcal{S}$ a positive literal a_Y. In implicational form, a definite Horn clause is equivalent to,

$$\bigwedge_{Y \in \mathcal{H}} a_Y \implies a_Z, \tag{2}$$

for some $\mathcal{H} \subseteq \mathcal{S}$ and $Z \in \mathcal{S} \setminus \mathcal{H}$ where $\mathcal{H}$ may be empty. If we associate a_Z with selecting Z for the cover, (2) encodes the constraint that selecting all of $\mathcal{H}$ for the cover requires also selecting Z. The *width* of a Horn clause is the number of literals, so a definite Horn clause of width two or less is such that $|\mathcal{H}| \leq 1$. Constraining exact problems solvable by LLP algorithms by clauses has been investigated in prior work [34] since they are naturally lattice-linear: (2) is false only if Z is not in the cover while $\mathcal{H}$ is a subset of the cover. In this case, Z is forbidden as no greater satisfying solution can be found until Z is included.

Because conjunctions of lattice-linear predicates remain lattice-linear, the LLP method has solved exact problems under constraints encoded by lattice-linear predicates [15, 34]. However, this principle does not transfer cleanly to LLP algorithms for approximate solutions. In this section, we use definite Horn constraints as a case study. Specifically, we modify the LLP method from Section 4 to achieve an f-approximation under definite Horn constraints of width two, and prove obtaining any constant-factor approximation to unweighted vertex cover under width-three Horn constraints is NP-hard.

5.1 Primal-Dual LLP Method for Width-Two Horn Constraints

Suppose we are given a collection of width-two Horn clauses as constraints. For simplicity and space constraints, we exclude width-one Horn clauses from our examination. Encode the constraints by the set $I \subseteq \mathcal{S} \times \mathcal{S}$, so that $(Y, Z) \in I$ means we must satisfy $a_Y \implies a_Z$. This defines the program,

$$\begin{aligned} \min \quad & w^\intercal v, \\ \text{s.t.} \quad & Av \geq \mathbf{1}, \\ & v_Z - v_Y \geq 0, \quad \forall (Y, Z) \in I, \\ & v \in \{0,1\}^{\mathcal{S}}, \end{aligned}$$

where $A = [\chi_{Y_1} \ \dots \ \chi_{Y_m}]$ is the matrix whose i^{th} column is the characteristic vector χ_{Y_i} of the set $Y_i \in \mathcal{S}$. After relaxing the final constraint to $v \geq \mathbf{0}$, we obtain the dual linear program,

$$\begin{aligned} \max \quad & \sum_{x \in \mathcal{U}} G[x], \\ \text{s.t.} \quad & price_G(Y) + \sum_{W:(W,Y) \in I} \lambda(W, Y) - \sum_{Z:(Y,Z) \in I} \lambda(Y, Z) \leq w(Y), \quad \forall Y \in \mathcal{S}, \\ & G, \lambda \geq 0. \end{aligned}$$

Unlike unconstrained weighted set cover this dual is not positive, and so a price raising mechanism will not immediately apply. To solve this, the next lemma describes a projection of the dual onto the G-coordinates using the closures of the implication graph corresponding to the Horn constraints. Anticipating this, observe that I defines a binary relation over $\mathcal{S}$, so we may consider the implication graph $(\mathcal{S}, I)$. Following Picard [31], a collection of sets $\mathcal{C} \subseteq \mathcal{S}$ is a *closure* if it is closed under the graph's reachability relation; equivalently $Y \in \mathcal{C}$ and $(Y, Z) \in I$ implies $Z \in \mathcal{C}$. We use closures to define an equivalent program.

Lemma 3. *A price vector $G \in \mathbb{R}^{\mathcal{U}}_{\geq 0}$ extends to a feasible dual solution (G, λ) if and only if $\sum_{Y \in \mathcal{C}} price_G(Y) \leq \sum_{Y \in \mathcal{C}} w(Y)$ for all closures $\mathcal{C} \subseteq \mathcal{S}$.*

The argument interprets λ as a flow in the reversed implication graph. To streamline the symbolic expressions we let $r_G(Y) \triangleq w(Y) - price_G(Y)$ be the residual budget for the set Y. The dual constraint is equivalent to the net flow through a vertex $Y \in \mathcal{S}$ being bounded by $r_G(Y)$. So we apply Gale's theorem (in a form translated to directed graphs).

Theorem 4 (Gale's feasibility theorem [8]). *Let (V, E) be a directed graph with capacities $c : E \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ and demands $d : V \to \mathbb{R}$. Moreover, let $\delta^-(v)$ and $\delta^+(v)$ be the incoming and outgoing edges for a vertex $v \in V$. There exists a flow $f : E \to \mathbb{R}_{\geq 0}$ satisfying $f(e) \leq c(e)$ for all $e \in E$ and,*

$$\sum_{e \in \delta^-(v)} f(e) - \sum_{e' \in \delta^+(v)} f(e') \geq d(v), \quad \forall v \in V,$$

if and only if $\sum_{v \in W} d(v) \leq \sum_{e \in \delta^-(W)} c(e)$ for all $W \subseteq V$.

Proof (of Lemma 3). Consider the reversed graph $(\mathcal{S}, I)^{\text{opp}}$ and assign infinite capacity to every edge. Let $\delta^-(Y)$ and $\delta^+(Y)$ be the incoming and outgoing edges of $Y \in \mathcal{S}$ respectively in the reversed graph. Then the dual constraint is equivalent to $\sum_{e \in \delta^-(Y)} \lambda(e) - \sum_{e' \in \delta^+(Y)} \lambda(e') \geq -r_G(Y)$ for all $Y \in \mathcal{S}$, so we let $-r$ give the demands. However, for any $\mathcal{C} \subseteq \mathcal{S}$, our construction makes,

$$\sum_{e \in \delta^-(\mathcal{C})} c(e) = \begin{cases} 0, & \text{if } \delta^-(\mathcal{C}) = \varnothing, \\ \infty, & \text{otherwise,} \end{cases}$$

where $\delta^-(\mathcal{C})$ is the natural extension of δ^- giving the edges incoming to $\mathcal{C}$ in $(\mathcal{S}, I)^{\text{opp}}$. However, $\mathcal{C}$ has no incoming edges in the reversed graph if and only if it is a closure in $(\mathcal{S}, I)$. And so, by Gale's theorem, there exists $\lambda \in \mathbb{R}^I_{\geq 0}$ such that $\sum_{e \in \delta^-(Y)} \lambda(e) - \sum_{e' \in \delta^+(Y)} \lambda(e') \geq -r_G(Y)$ if and only if,

$$0 \geq -\sum_{Y \in \mathcal{C}} r_G(Y) = \sum_{Y \in \mathcal{C}} \big(price_G(Y) - w(Y)\big),$$

for all closures $\mathcal{C}$ in $(\mathcal{S}, I)$. So, rearranging terms gives the result. □

Now suppose G is feasible in the sense of Lemma 3. If one were to raise the value of $G[x]$ by value $\delta > 0$, then G remains feasible if and only if,

$$\delta|\{Y \in \mathcal{C} : x \in Y\}| + \sum_{Y \in \mathcal{C}} \mathit{price}_G(Y) \leq \sum_{Y \in \mathcal{C}} w(Y),$$

for all closures $\mathcal{C}$. Therefore, the largest possible such increase is,

$$\Delta_G(x) \triangleq \min_{\mathcal{C}:\exists Y \in \mathcal{C}: x \in Y} \frac{\sum_{Y \in \mathcal{C}} r_G(Y)}{|\{Y \in \mathcal{C} : x \in Y\}|}.$$

We obtain a (possibly non-optimal) solution to the projected program by computing a price vector satisfying,

$$B_{\text{hcwsc}}(G) \equiv \forall x \in \mathcal{U} : \Delta_G(x) = 0,$$

and so we let $\mathit{forbidden}_G \equiv \mathit{clslackmin}_G$ with,

$$\begin{aligned} \mathit{clslackmin}_G(x) \equiv\ & \Delta_G(x) > 0, \\ & \wedge \forall z \in \mathcal{U} : z \leftrightsquigarrow x \wedge \Delta_G(z) > 0 \implies x \leq_{\text{lex}} z, \end{aligned}$$

where the (reflexive) neighboring relation $z \leftrightsquigarrow x$ is defined by x and z being contained in sets in a shared weak component of $(\mathcal{S}, I)$. The neighboring relation is used in the proof of Lemma 4 to ensure that forbidden elements stay forbidden when other elements are advanced, and that advancing a forbidden element does not create a state outside of our chosen lattice. Specifically, two distinct cl-slack-minimal elements are never contained in sets in a shared component of the implication graph, and so simultaneous advances affect disjoint component-contained closure systems and preserve feasibility. Finally, we advance the state $G[x]$ via the update $G[x] \leftarrow G[x] + \Delta_G(x)$. Efficiently computing Δ_G (and corresponding forbidden and advancements) is done using parametric min-cut algorithms [9]; we defer this explanation to the end of the section. For a parallel or distributed implementation, we remark that computing $\Delta_G(x)$ is local to the components containing a set containing x.

In a similar vein to Section 4, we let G^* be the solution constructed by the following sequential process: Initially let $G^*_0 = \mathbf{0}$, and first select the lex-minimal z_1 such that $\Delta_{G^*_0}(z_1) > 0$. Update via the advancement $G^*_1[z_1] := G^*_0[z_1] + \Delta_{G^*_0}(z_1)$. Then select the next lex-minimal z_2 with $\Delta_{G^*_1}(z_2) > 0$, and perform the same advancement to obtain G^*_2. Repeat until $\Delta_{G^*_t}(x) = 0$ for all $x \in \mathcal{U}$, and take $G^* = G^*_t$. Observe that G^* is feasible for the projected program defined by Lemma 3 by the advances. Finally, define the lattice,

$$\mathcal{L}_{\text{hcwsc}} \triangleq \{G \in \mathbb{R}^{\mathcal{U}}_{\geq 0} : G \leq G^*\}.$$

We show B_{hcwsc} is lattice-linear on $\mathcal{L}_{\text{hcwsc}}$, and verify the advancements never leave the lattice.

Lemma 4. *Select $G \in \mathcal{L}_{\text{hcwsc}}$. If $\neg B_{hcwsc}(G)$, then all of the following are true:*

(i) *some element $x \in \mathcal{U}$ is cl-slack-minimal,*
(ii) *advancing $G[x] := G[x] + \Delta_G(x)$ remains in $\mathcal{L}_{\text{hcwsc}}$,*
(iii) *for all $H \in \mathcal{L}_{\text{hcwsc}}$, if $H \geq G$ and $H[x] = G[x]$ then $\neg B_{\text{hcwsc}}(H)$.*

Proof. For the first item, see that $\neg B_{\text{hcwsc}}(G)$ implies $\Delta_G(x) > 0$ for some $x \in \mathcal{U}$, and so some such element must be lex-minimal among any neighbors z with $\Delta_G(z) > 0$. This shows (i). Now, select a cl-slack-minimal x. For (ii) we proceed by contradiction: assume $G^*[x] < G[x] + \Delta_G(x)$. Observe that every element is contained in a set in some tight closure by the construction of G^*. We can localize this to the components of the implication graph. In particular, let $\{K_j\}_j$ be the (weak) components of $(\mathcal{S}, I)$, and observe the intersection $\mathcal{C} \cap K_j$ is also a closure for all j and closures $\mathcal{C}$. Then, for any element $x \in \mathcal{U}$ and closure $\mathcal{C}$ such that there exists a set $Y \in \mathcal{C}$ with $x \in Y$,

$$\frac{\sum_{Y \in \mathcal{C}} r_G(Y)}{|\{Y \in \mathcal{C} : x \in Y\}|} = \frac{\sum_j \sum_{Y \in \mathcal{C} \cap K_j} r_G(Y)}{\sum_j |\{Y \in \mathcal{C} \cap K_j : x \in Y\}|},$$
$$\geq \min_{j : \exists Y \in \mathcal{C} \cap K_j : x \in Y} \frac{\sum_{Y \in \mathcal{C} \cap K_j} r_G(Y)}{|\{Y \in \mathcal{C} \cap K_j : x \in Y\}|},$$

as graph components are disjoint. This implies that the minimization defining $\Delta_G(x)$ is attained by a closure contained entirely in a component. And so, if $\Delta_G(x) = 0$ then some closure containing a set containing x is both tight and contained entirely in a single graph component.

Since $\Delta_{G^*}(x) = 0$, there exists a tight closure $\mathcal{C}$ in G^* entirely contained in a component, which contains a set containing x. Let $\widehat{G}$ be the state resulting from the advancement $G[x] := G^*[x]$; we see $\mathcal{C}$ is slack in $\widehat{G}$. So because $\widehat{G}[x] = G^*[x]$, there exists $z \neq x$ contained in a set in $\mathcal{C}$ (i.e. $z \leftrightsquigarrow x$) such that $G^*[z] > \widehat{G}[z] = G[z]$. Hence we may select the earliest selected distinct neighbor $z_i \neq x$ in the construction of G^* for which $G^*[z_i] > G[z_i]$. Let $P \triangleq G^*_{i-1}$. By the minimality of i, it follows that $P[z'] \leq \widehat{G}[z']$ for all neighbors $z' \leftrightsquigarrow x$. Therefore, for every closure $\mathcal{C}$ contained in a component for which there exists $Y \in \mathcal{C}$ with $x \in Y$,

$$\sum_{Y \in \mathcal{C}} price_P(Y) \leq \sum_{Y \in \mathcal{C}} price_{\widehat{G}}(Y).$$

Since $\widehat{G}[x] < G[x] + \Delta_G(x)$ we have $\Delta_{\widehat{G}}(x) > 0$, and so by the previous equation it follows that $\Delta_P(x) \geq \Delta_{\widehat{G}}(x) > 0$. That is, no closure containing a set containing x was tight when z_i was chosen for G^*, so it must be that $z_i <_{\text{lex}} x$. But, because $z_i \leftrightsquigarrow x$ and x is cl-slack-minimal in G, we observe $\Delta_G(z_i) = 0$. So there exists a closure $\mathcal{C}'$ such that for some $Y \in \mathcal{C}'$ we have $z_i \in Y$, and $\sum_{Y \in \mathcal{C}'} price_G(Y) = \sum_{Y \in \mathcal{C}'} w(Y)$. However, because $G \leq G^*$ and $G[z_i] < G^*[z_i]$ it is also true that,

$$\sum_{Y \in \mathcal{C}'} price_G(Y) < \sum_{Y \in \mathcal{C}'} price_{G^*}(Y).$$

This makes $\sum_{Y \in \mathcal{C}'} price_{G^*}(Y) > \sum_{Y \in \mathcal{C}'} w(Y)$ which contradicts the feasibility of G^* under Lemma 3. This gives (ii).

Algorithm LLP-HCWeightedSetCover: Parallel f-approximation for weighted set cover with width-two Horn constraints.

Input: Universe $\mathcal{U}$; collection $\mathcal{S}$; weights $w : \mathcal{S} \to \mathbb{R}_{\geq 0}$; implications $I \subseteq \mathcal{S} \times \mathcal{S}$
Output: $\mathcal{C} \subseteq \mathcal{S}$

G: array[$\mathcal{U}$] of reals, initially $\forall x \in \mathcal{U} :\ G[x] := 0$ `// element prices`
$V := \mathcal{S} \cup \{s, t\}$
$E := I \cup (\{s\} \times \mathcal{S}) \cup (\mathcal{S} \times \{t\})$

$$c_0(Y, Z) := \begin{cases} \max\{0, -r_G(Z)\}, & \text{if } s = Y, \\ \max\{0, r_G(Y)\}, & \text{if } t = Z, \\ \infty, & \text{otherwise.} \end{cases}$$

forbidden (x) : $clslackmin_G(x)$
$\Rightarrow$ **advance** : $G[x] := G[x] + \Delta_G(x)$
$(A, T) :=$ maximal (source-side inclusion) min-cut of (V, E) with c_0
return $C := A \setminus \{s\}$

Finally, for (iii) select $H \in \mathcal{L}_{hcwsc}$ such that $H \geq G$ and $H[x] = G[x]$. So long as $\Delta_H(x) > 0$, the element x is also cl-slack-minimal in H by $H \geq G$ (as it can be observed that $G \mapsto \Delta_G(x)$ is antitone). By (ii), we find $H[x] < G^*[x]$, and so by way of $H \leq G^*$,

$$\sum_{Y \in \mathcal{C}} price_H(Y) < \sum_{Y \in \mathcal{C}} price_{G^*}(Y),$$

for all closures $\mathcal{C}$ containing a set containing x. This implies that every closure containing a set containing x must be slack in H, equivalently $\Delta_H(x) > 0$. This concludes (iii). □

The algorithm is given in LLP-HCWeightedSetCover. Notice $\Delta_G(x)$ remains zero after x is advanced, so the algorithm terminates after at most n advancements. The solution is extracted via computing a maximal min-cut with respect to source-side inclusion (which is found efficiently via reachability computations on the residual graph; see [6]) on the following construction: create a flow network (V, E) from $(\mathcal{S}, I)$ by introducing source and sink vertices s and t with edges to every element in $\mathcal{S}$. We use the capacities,

$$c_0(s, Y) = \max\{0, -r_G(Y)\}, \qquad c_0(Y, t) = \max\{0, r_G(Y)\},$$

and infinity for the preexisting edges from the implications in I. Therefore, any cut (A, T) whose source side is not given by $A = \mathcal{C} \cup \{s\}$ for some closure $\mathcal{C}$ has infinite capacity, so any min-cut always corresponds to a closure in this way. This is a standard min-cut formulation of closure problems given in classic works [31, 27, 9]. Now, if we let $\kappa_G(\mathcal{C})$ be the capacity of $(\mathcal{C} \cup \{s\}, \{t\} \cup \mathcal{S} \setminus \mathcal{C})$ for a

closure $\mathcal{C}$, after noting that the empty set is always a closure we see,

$$\begin{aligned}\kappa_G(\mathcal{C}) &= \left(\sum_{Y \notin \mathcal{C}} \max\{0, -r_G(Y)\}\right) + \left(\sum_{Y \in \mathcal{C}} \max\{0, r_G(Y)\}\right), \\ &= \left(\sum_{Y \in \mathcal{S}} \max\{0, -r_G(Y)\}\right) + \left(\sum_{Y \in \mathcal{C}} \max\{0, r_G(Y)\} - \max\{0, -r_G(Y)\}\right), \\ &= \kappa_G(\varnothing) + \sum_{Y \in \mathcal{C}} r_G(Y).\end{aligned}$$

Since $\sum_{Y \in \mathcal{C}} r_G(Y) \geq 0$ for all closures $\mathcal{C}$ for feasible price vectors G by way of Lemma 3, it follows that a closure generates a min-cut if and only if its capacity equals $\kappa_G(\varnothing)$, equivalently if and only if $\sum_{Y \in \mathcal{C}} r_G(Y) = 0$ (i.e. $\mathcal{C}$ is tight).

Note that source-side sets of min-cuts are closed under union [32], so the preceding analysis shows the maximal min-cut $(\mathcal{C} \cup \{s\}, \{t\} \cup \mathcal{S} \setminus \mathcal{C})$ is such that $\mathcal{C}$ is the union of all tight closures and $\mathcal{C}$ is tight itself since it corresponds to a min-cut. This fact is used to show the algorithm returns a feasible set cover, and (weak) duality is used to obtain the approximation factor.

Theorem 5. *Let $f := \max_{x \in \mathcal{U}} |\{Y \in \mathcal{S} : x \in Y\}|$ be the maximum element frequency. LLP-HCWeightedSetCover returns a feasible set cover $\mathcal{C}$ with $\sum_{Y \in \mathcal{C}} w(Y) \leq f \cdot \mathrm{OPT}$.*

Proof. At termination $B_{hcwsc}(G)$ holds true, so $\Delta_G(x) = 0$ for all $x \in \mathcal{U}$. It follows that there exists some tight closure $\mathcal{C}'$ such that there exists $Y \in \mathcal{C}'$ with $x \in Y$, and so because our prior discussion shows $\mathcal{C}$ contains all tight closures, we have $Y \in \mathcal{C}' \subseteq \mathcal{C}$. As the choice of $x \in \mathcal{U}$ was arbitrary, $\mathcal{C}$ defines a set cover. Moreover, $\mathcal{C}$ is a closure in $(\mathcal{S}, I)$, and so $\mathcal{C}$ satisfies the Horn clauses. Now observe that the value of the cover gives $\sum_{Y \in \mathcal{C}} w(Y) = \sum_{Y \in \mathcal{C}} \mathit{price}_G(Y)$ because $\mathcal{C}$ is tight (as it corresponds to the min-cut, the maximal one). But,

$$\sum_{Y \in \mathcal{C}} \mathit{price}_G(Y) = \sum_{Y \in \mathcal{C}} \sum_{x \in Y} G[x] \leq f \sum_{x \in \mathcal{U}} G[x].$$

Because G extends to a feasible solution of the dual relaxation by Lemma 3, weak duality makes $\sum_{x \in \mathcal{U}} G[x] \leq \mathrm{OPT}$. And so $\sum_{Y \in \mathcal{C}} w(Y) \leq f \cdot \mathrm{OPT}$. □

To conclude the section we discuss how forbidden and advance can be computed efficiently. The neighbor relation $\leftrightsquigarrow$ can be precomputed by identifying the components and associating each $x \in \mathcal{U}$ with the components (see [14] for LLP algorithms for connected components) containing a set containing x. So, what remains for verifying we may efficiently evaluate $\mathit{forbidden}_G(x)$ and advance is giving a method for evaluating Δ_G. One may evaluate $\Delta_G(x)$ via pre-existing methods: Again create a flow network from $(\mathcal{S}, I)$ by introducing source and sink vertices s and t with edges to every element in $\mathcal{S}$, but now assign the parameterized capacities for each $Y \in \mathcal{S}$,

$$c_\theta(s, Y) = \theta \chi_Y(x) + \max\{0, -r_G(Y)\}, \qquad c_\theta(Y, t) = \max\{0, r_G(Y)\},$$

and infinity for the remaining edges defined by I.

Again, any cut whose source side is not given by $\mathcal{C} \cup \{s\}$ where C is a closure has infinite capacity, so every min-cut is a closure. The capacity $\kappa_G(\mathcal{C})$ defined by a closure $\mathcal{C}$ is exactly,

$$\kappa_G(\mathcal{C}) = \left(\sum_{Y \notin \mathcal{C}} \theta \chi_Y(x) + \max\{0, -r_G(Y)\}\right) + \left(\sum_{Y \in \mathcal{C}} \max\{0, r_G(Y)\}\right),$$

$$= \kappa_G(\varnothing) + \left(\sum_{Y \in \mathcal{C}} r_G(Y)\right) - \theta |\{Y \in \mathcal{C} : x \in Y\}|, \quad (3)$$

where $\kappa_G(\varnothing) = \sum_{Y \in \mathcal{S}} \theta \chi_Y(x) + \max\{0, -r_G(Y)\}$ is the capacity of the cut defined by the empty closure. Note that because G is feasible, by Lemma 3 we have $\sum_{Y \in \mathcal{C}} r_G(Y) \geq 0$ for all closures $\mathcal{C}$. So, (3) shows that the cut defined by the empty closure is a min-cut exactly while $\theta \leq \Delta_G(x)$. At $\theta = \Delta_G(x)$, the capacity of the cut generated by a nonempty closure first ties that of the empty closure and becomes strictly cheaper thereafter. Hence, $\Delta_G(x)$ defines the smallest breakpoint in our parametric min-cut instance, and so is computable by Gallo, Grigoriadis, and Tarjan [9] as the source-side capacities are increasing in θ and all remaining edges are constant.

5.2 Hardness for Width-Three Horn Constraints

Now we show it is NP-hard to achieve any constant-factor approximation to (unit-weight) vertex cover with Horn constraints of width three. Recalling this is a weighted set cover problem through the edge-incidence sets of the vertices, this and the last section show a boundary at width-two Horn constraints.

Let $\bigwedge_{j=1}^{m} C_j$ be a 3-SAT formula with m clauses over n variables $b_1, \ldots, b_n$, and $M \triangleq \lceil \alpha(n+m) \rceil + 1$ for a fixed constant α. In the reduction, α will encode the target approximation factor for a Horn-constrained vertex cover algorithm. Create a graph (V, E) as follows: For each variable b_i, associate two vertices t_i and f_i with an edge between them, and for each clause C_j create a set of $M+1$ isolated vertices $Q_j \triangleq \{q_{0,j}, q_{1,j}, \ldots, q_{M,j}\}$. For a literal ℓ, define the function,

$$\phi(\ell) \triangleq \begin{cases} f_i, & \text{if } \ell = b_i, \\ t_i, & \text{if } \ell = \neg b_i. \end{cases}$$

That is, when ℓ is defined from the variable b_i, $\phi(\ell)$ maps to f_i when the literal is positive and t_i when the literal is negative. Suppose each clause is written as $C_j = \ell_{1,j} \vee \ell_{2,j} \vee \ell_{3,j}$. To make an instance with constraints, add the Horn rule,

$$(\phi(\ell_{1,j}) \in \mathcal{C}) \wedge (\phi(\ell_{2,j}) \in \mathcal{C}) \implies q_{0,j} \in \mathcal{C}, \quad (4)$$

and the M separate Horn rules,

$$(q_{0,j} \in \mathcal{C}) \wedge (\phi(\ell_{3,j}) \in \mathcal{C}) \implies q_{i,j} \in \mathcal{C}, \quad (5)$$

for each clause C_j and $i \in \{1, \ldots, M\}$. Finally, the reduction is polynomial because α is fixed and independent of n and m and $M = O(n + m)$.

Lemma 5. *If the 3-SAT instance is satisfiable, then there exists a feasible Horn-constrained vertex cover $\mathcal{C}$ of size $|\mathcal{C}| \leq n + m$.*

Proof. Examine any satisfying assignment. We construct a feasible vertex cover $\mathcal{C}$ in the following way: If b_i is assigned true in the assignment, then add t_i to the cover. Otherwise, add f_i to the cover. Then, add $q_{0,j}$ to the cover for any clause C_j whose first two literals are false to complete $\mathcal{C}$.

See that $\phi(\ell) \in \mathcal{C}$ if and only if the literal ℓ is false in the 3-SAT assignment. So our rule for adding $q_{0,j}$ easily satisfies (4) for each clause C_j. Furthermore, since we started with a satisfying assignment, $q_{0,j} \in \mathcal{C}$ implies the third literal $\ell_{3,j}$ must be true for C_j to be satisfied, meaning that $\phi(\ell_{3,j}) \notin \mathcal{C}$ in this case. Therefore, for each clause all M implications in (5) are satisfied. This makes $\mathcal{C}$ a vertex cover satisfying the Horn constraints. The construction makes $|\mathcal{C}| \leq n+m$, so the lemma follows. □

Lemma 6. *If there exists a feasible Horn-constrained vertex cover $\mathcal{C}$ of size $|\mathcal{C}| < M$, then the 3-SAT instance is satisfiable.*

Proof. Since $\mathcal{C}$ covers every edge (t_i, f_i), we may define an assignment by,

$$b_i := \begin{cases} \text{true}, & \text{if } f_i \notin \mathcal{C}, \\ \text{false}, & \text{if } t_i \notin \mathcal{C}, \\ \text{arbitrary}, & \text{otherwise.} \end{cases}$$

Under this assignment, $\phi(\ell) \notin \mathcal{C}$ implies the literal ℓ is true. Now examine the clause C_j. If either $\phi(\ell_{1,j}) \notin \mathcal{C}$ or $\phi(\ell_{2,j}) \notin \mathcal{C}$ then the corresponding literal is true (so C_j is satisfied). Otherwise, if $\{\phi(\ell_{1,j}), \phi(\ell_{2,j})\} \subseteq \mathcal{C}$, then as $\mathcal{C}$ is feasible (4) forces $q_{0,j} \in \mathcal{C}$. However, $\phi(\ell_{3,j})$ cannot be contained in $\mathcal{C}$ as then the M implications in (5) force $\{q_{1,j}, \ldots, q_{M,j}\} \subseteq \mathcal{C}$, contradicting $|\mathcal{C}| < M$. So $\ell_{3,j}$ is true. Thus, C_j is satisfied in any case, and as our choice of j was arbitrary the instance is satisfiable. □

Theorem 6. *For every fixed constant $\alpha \geq 1$, unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial-time α-approximation algorithm for (unit-weight) vertex cover subject to width-three Horn constraints.*

Proof. Examine any 3-SAT instance, and let $\mathcal{C}$ be any α-approximate solution over (V, E) with the constraints we've described. If the 3-SAT instance is satisfiable, then Lemma 5 implies $\text{OPT} \leq n + m$, so,

$$|\mathcal{C}| \leq \alpha \text{OPT} \leq \alpha(n + m) < M.$$

If the 3-SAT instance is not satisfiable, then the contrapositive of Lemma 6 implies $|\mathcal{C}| \geq M$. Consequently, the 3-SAT instance is satisfiable if and only if $|\mathcal{C}| < M$. And so, if there did exist an α-approximation algorithm to width-three Horn-constrained vertex cover, we could solve 3-SAT. □

References

1. Alves, D.R., Garg, V.K.: Parallel minimum spanning tree algorithms via lattice linear predicate detection. In: IEEE International Parallel and Distributed Processing Symposium, IPDPS Workshops 2022, Lyon, France, May 30 - June 3, 2022. pp. 774–782. IEEE (2022). https://doi.org/10.1109/IPDPSW55747.2022.00131
2. Alves, D.R., Garg, V.K.: A common parallel framework for LLP combinatorial problems. arXiv preprint arXiv:2603.13147 (2026)
3. Bar-Yehuda, R., Even, S.: A linear-time approximation algorithm for the weighted vertex cover problem. Journal of Algorithms **2**(2), 198–203 (1981)
4. Bar-Yehuda, R., Even, S.: A local-ratio theorem for approximating the weighted vertex cover problem. Annals of Discrete Mathematics **25**, 27–46 (1985)
5. Chvátal, V.: A greedy heuristic for the set-covering problem. Mathematics of Operations Research **4**(3), 233–235 (1979)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge, MA, 3rd edn. (2009)
7. Dinur, I., Safra, S.: On the hardness of approximating minimum vertex cover. Annals of Mathematics **162**(1), 439–485 (2005)
8. Gale, D.: A theorem on flows in networks. In: Classic Papers in Combinatorics, pp. 259–268. Springer (1957)
9. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM Journal on Computing **18**(1), 30–55 (1989)
10. Garg, V.K.: Lattice Theory with Computer Science Applications. Wiley, New York, NY (2015)
11. Garg, V.K.: Predicate detection to solve combinatorial optimization problems. In: Scheideler, C., Spear, M. (eds.) SPAA '20: 32nd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, July 15-17, 2020. pp. 235–245. ACM (2020). https://doi.org/10.1145/3350755.3400235
12. Garg, V.K.: A lattice linear predicate parallel algorithm for the housing market problem. In: Johnen, C., Schiller, E.M., Schmid, S. (eds.) Stabilization, Safety, and Security of Distributed Systems - 23rd International Symposium, SSS 2021, Virtual Event, November 17-20, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13046, pp. 108–122. Springer (2021). https://doi.org/10.1007/978-3-030-91081-5_8
13. Garg, V.K.: A lattice linear predicate parallel algorithm for the dynamic programming problems. In: Proc. of the Int'l Conf. on Distributed Computing and Networking (ICDCN). Springer-Verlag, Delhi, India (2022)
14. Garg, V.K.: A systematic approach to parallel algorithms. `https://users.ece.utexas.edu/~garg/paralgo.html` (2026)
15. Garg, V.K.: Keynote talk: Lattice linear predicate algorithms for the constrained stable marriage problem with ties. In: 24th International Conference on Distributed Computing and Networking, ICDCN 2023, Kharagpur, India, January 4-7, 2023. pp. 2–11. ACM (2023). https://doi.org/10.1145/3571306.3571318
16. Goemans, M.X., Williamson, D.P.: The primal-dual method for approximation algorithms and its application to network design problems. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-Hard Problems, pp. 144–191. PWS Publishing (1997)
17. Gupta, A.T., Kulkarni, S.S.: Extending lattice linearity for self-stabilizing algorithms. In: International Symposium on Stabilizing, Safety, and Security of Distributed Systems. pp. 365–379. Springer (2021)

18. Gupta, A.T., Kulkarni, S.S.: Eventually lattice-linear algorithms. Journal of Parallel and Distributed Computing **185**, 104802 (2024)
19. Gupta, A.T., Kulkarni, S.S.: Tolerance to asynchrony of an algorithm for gathering myopic robots on an infinite triangular grid. In: 2024 19th European Dependable Computing Conference (EDCC). pp. 61–68. IEEE (2024)
20. Gupta, A.T., Kulkarni, S.S.: Tolerance to asynchrony in algorithms for multiplication and modulo. Theoretical Computer Science **1024**, 114914 (2025)
21. Gusfield, D., Irving, R.W.: The stable marriage problem: structure and algorithms. MIT Press (1989)
22. Hochbaum, D.S.: Approximation algorithms for the set covering and vertex cover problems. SIAM Journal on Computing **11**(3), 555–556 (1982)
23. Israeli, A., Itai, A.: A fast and simple randomized parallel algorithm for maximal matching. Information Processing Letters **22**(2), 77–80 (1986)
24. Johnson, D.S.: Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences **9**(3), 256–278 (1974)
25. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2-\varepsilon$. Journal of Computer and System Sciences **74**(3), 335–349 (2008)
26. Khuller, S., Vishkin, U., Young, N.E.: A primal-dual parallel approximation technique applied to weighted set and vertex covers. Journal of Algorithms **17**(2), 280–289 (1994)
27. Lawler, E.L.: Sequencing jobs to minimize total weighted completion time subject to precedence constraints. In: Annals of Discrete Mathematics, vol. 2, pp. 75–90. Elsevier (1978)
28. Lovász, L.: On the ratio of optimal integral and fractional covers. Discrete Mathematics **13**(4), 383–390 (1975)
29. Luby, M.: A simple parallel algorithm for the maximal independent set problem. In: Proceedings of the seventeenth annual ACM symposium on Theory of computing. pp. 1–10 (1985)
30. McCormick, S.T., Peis, B., Verschae, J., Wierz, A.: Primal–dual algorithms for precedence constrained covering problems. Algorithmica **78**(3), 771–787 (2017)
31. Picard, J.C.: Maximal closure of a graph and applications to combinatorial problems. Management Science **22**(11), 1268–1272 (1976)
32. Picard, J.C., Queyranne, M.: On the structure of all minimum cuts in a network and applications. Mathematical Programming Studies **13**, 8–16 (1980)
33. Rajagopalan, S., Vazirani, V.V.: Primal-dual RNC approximation algorithms for set cover and covering integer programs. SIAM Journal on Computing **28**(2), 525–540 (1998)
34. Streit, R., Garg, V.K.: Constrained cuts, flows, and lattice-linearity. arXiv preprint arXiv:2512.18141 (2025)
35. Vazirani, V.V.: Approximation Algorithms. Springer-Verlag, Berlin, Germany (2001)
36. Williamson, D.P., Shmoys, D.B.: The Design of Approximation Algorithms. Cambridge University Press (2011)