New Sequence-Independent Lifting Techniques for Cover Inequalities and When They Induce Facets

Siddharth Prasad[1], **Ellen Vitercik**[2], **Maria-Florina Balcan**[1] and **Tuomas Sandholm**[1,3,4,5]
[1]School of Computer Science, Carnegie Mellon University
[2]Department of Management Science & Engineering, Stanford University
[3]Optimized Markets, Inc.
[4]Strategy Robot, Inc.
[5]Strategic Machine, Inc.
{sprasad2, ninamf, sandholm}@cs.cmu.edu, vitercik@stanford.edu

Abstract

Sequence-independent lifting is a procedure for strengthening valid inequalities of an integer program. We generalize the sequence-independent lifting method of Gu, Nemhauser, and Savelsbergh (GNS lifting) for cover inequalities and correct an error in their proposed generalization. We obtain a new sequence-independent lifting technique—*piecewise-constant (PC) lifting*—with a number of important properties. We derive a broad set of sufficient conditions under which PC lifting yields facets—the first characterization of facet-defining sequence-independent liftings that are efficiently computable from the underlying cover. Finally, we demonstrate via experiments that PC lifting can be a useful alternative to GNS lifting. We test PC lifting atop a number of novel cover inequality generation routines, which prove to be effective in experiments with CPLEX. PC lifting delivers strong numerical properties making it practically relevant for integer programming solvers.

1 Introduction

Lifting is a technique for strengthening cutting planes for integer programs by increasing the coefficients of variables that are not in the cut. We study lifting methods for valid cuts of *knapsack polytopes*, which have the form $\text{conv}(P)$ where

$$P = \left\{ \boldsymbol{x} \in \{0,1\}^n : \sum_{j=1}^{n} a_j x_j \leq b \right\}$$

for $a_1, \ldots, a_n, b \in \mathbb{N}$ with $0 < a_1, \ldots, a_n \leq b$. We interpret P as the set of feasible packings of n items with weights $a_1, \ldots, a_n$ into a knapsack of capacity b. Such *knapsack constraints* arise in binary integer programs from various industrial applications such as resource allocation, auctions, and container packing. They are a very general and expressive modeling tool, as any linear constraint involving binary variables admits an equivalent knapsack constraint by replacing negative-coefficient variables with their complements. A *minimal cover* is a set $C \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in C} a_j > b$ and $\sum_{j \in C \setminus \{i\}} a_j \leq b$ for all $i \in C$. That is, the items in C cannot all fit in the knapsack, but any proper subset of C can. The *minimal cover inequality/cut* corresponding to C is the inequality

$$\sum_{j \in C} x_j \leq |C| - 1,$$

which enforces that the items in C cannot all be selected. A *lifting* of the minimal cover inequality is any valid inequality of the form

$$\sum_{j \in C} x_j + \sum_{j \notin C} \alpha_j x_j \leq |C| - 1. \tag{1}$$

The lifting coefficients α_j are often computed one-by-one—a process called *sequential lifting* that depends on the lifting order. Sequential lifting can be expensive since one must solve an optimization problem for each coefficient. Furthermore, one must reckon with the question of what lifting order to use. To lessen this computational burden, the lifting coefficients can be computed simultaneously. This method is called *sequence-independent lifting* and is the focus of this work. Our contributions include: (i) a generalization of the seminal sequence-independent lifting method developed by Gu et al. [2000] and a correction of their proposed generalization; (ii) the first broad conditions under which sequence-independent liftings that are efficiently computable from the underlying cover—via our new techniques—define facets of $\text{conv}(P)$ (facet-defining cuts are the gold standard for cutting planes since they are in a formal sense the strongest kind of cutting plane); and (iii) new cover inequality generation methods that, together with our new lifting techniques, display promising practical performance in experiments.

1.1 Preliminaries on Cutting Planes and Sequence-Independent Lifting

A *(valid) cutting plane* for P is any constraint $\boldsymbol{a}^\top \boldsymbol{x} \leq b$ that is satisfied by all $\boldsymbol{x} \in P$. Cutting planes are one of the most critical components of all modern state-of-the-art branch-and-cut integer programming solvers like Gurobi, CPLEX, Xpress, *etc.* They yield better dual bounds, thereby speeding up branch-and-cut tree search, by cutting off portions of the feasible region of the linear-programming relaxation (including

the optimal fractional solution) while preserving all integer feasible solutions. Lifted cover inequalities (1) are a specific family of cuts that are used by all the aforementioned solvers.

A cut $\boldsymbol{a}^\top \boldsymbol{x} \leq b$ is a *facet* of $\mathrm{conv}(P)$ if it defines a dimension-$(\dim(P)-1)$ face of $\mathrm{conv}(P)$. Facet-defining cuts are thus the strongest kind of cutting plane and provide the best dual bounds for use within branch-and-cut.

Lifting. We begin with an overview of the *lifting function* $f : [0, b] \to \mathbb{R}$ associated with a minimal cover C, defined by $f(z) = |C| - 1 - \max\{\sum_{j \in C} x_j : \sum_{j \in C} a_j x_j \leq b - z, x_j \in \{0, 1\}\}$. For $i \notin C$, the value $f(a_i)$ is the maximum possible coefficient α_i such that $\sum_{j \in C} x_j + \alpha_i x_i \leq |C| - 1$ is valid for $\mathrm{conv}(P)$. The lifting function has a more convenient closed form due to Balas [1975]. First, relabel the items so $C = \{1, \ldots, t\}$ and $a_1 \geq \cdots \geq a_t$. Let $\mu_0 = 0$ and for $h = 1, \ldots, t$ let $\mu_h = a_1 + \cdots + a_h$. Let $\lambda = \mu_t - b > 0$ be the cover's excess weight. Then, $f(z) = 0$ if $0 \leq z \leq \mu_1 - \lambda$ and $f(z) = h$ if $\mu_h - \lambda < z \leq \mu_{h+1} - \lambda$. The lifting function has an intuitive interpretation: $f(z)$ is the maximum h such that an item of weight z cannot be brought into in C and fit in the knapsack, even if we are allowed to discard any h items from C. The lifting function f may be used to maximally lift a *single* variable not in the cover. To lift a second variable, a new lifting function must be computed. This (order-dependent) process can be continued to lift all remaining variables, and is known as *sequential lifting*. Conforti et al. [2014] and Hojny et al. [2020] contain further details.

Superadditivity and Sequence-Independent Lifting. A function $g : D \to \mathbb{R}$ is superadditive if $g(u + v) \geq g(u) + g(v)$ for all $u, v, u + v \in D$. If $g \leq f$ is superadditive, $\sum_{j \in C} x_j + \sum_{j \notin C} g(a_j) x_j \leq |C| - 1$ is a valid *sequence-independent* lifting for $\mathrm{conv}(P)$. This result is due to Wolsey [1977]; Gu et al. [2000] generalize to mixed 0-1 integer programs. The lifting function f is generally not superadditive. Gu et al. [2000] construct a superadditive function $g \leq f$ as follows. Let $\rho_h = \max\{0, a_{h+1} - (a_1 - \lambda)\}$ be the excess weight of the cover if the heaviest item is replaced with a copy of the $(h+1)$-st heaviest item. For $h \in \{0, \ldots, t-1\}$, let $F_h = (\mu_h - \lambda + \rho_h, \mu_{h+1} - \lambda]$ and for $h \in \{1, \ldots, t-1\}$, let $S_h = (\mu_h - \lambda, \mu_h - \lambda + \rho_h]$. S_h is nonempty if and only if $\rho_h > 0$. For $w : [0, \rho_1] \to [0, 1]$, Gu et al. define $g_w(z) =$

$$\begin{cases} 0 & z = 0 \\ h & z \in F_h, h = 0, \ldots, t-1 \\ h - w(\mu_h - \lambda + \rho_h - z) & z \in S_h, h = 1, \ldots, t-1. \end{cases}$$

Gu et al. prove that for $w(x) = x/\rho_1$, g_w is superadditive. We call this particular lifting function the *Gu-Nemhauser-Savelsbergh (GNS) lifting function*. Furthermore, g_w is undominated, that is, there is no superadditive g' with $f \geq g' \geq g_w$ and $g'(z') > g_w(z')$ for some $z' \in [0, b]$.

1.2 Our Contributions

In Section 2, we prove that under a certain condition, g_w is superadditive for any linear symmetric function w. This generalizes Gu et al.'s [2000] result for $w(x) = x/\rho_1$ and furthermore corrects an error in their proposed generalization, which incorrectly claims w can be any symmetric function. Of particular interest is the constant function $w = 1/2$; we call the resulting lifting *piecewise-constant (PC) lifting*. In Section 2.1 we give a thorough comparison of PC and GNS lifting. We show that GNS lifting can be arbitrarily worse than PC lifting, and characterize the full domination criteria between the two methods. In Section 2.2, we provide a broad set of conditions under which PC lifting defines facets of $\mathrm{conv}(P)$. *To our knowledge, these are the first conditions for facet-defining sequence-independent liftings that are efficiently computable from the underlying cover.*[1] Furthermore, PC-lifted cuts only have integral and half-integral coefficients making them practically relevant for solvers. In Section 2.2 we give an example that shows PC lifting can be significantly stronger than another half-integral sequence-independent lifting procedure due to Letchford and Souli [2019] that is currently implemented in the FICO Xpress solver [Perregard, 2024].

In Section 3, we experimentally evaluate our lifting techniques in conjunction with a number of novel cover cut generation techniques. Our cut generation techniques do not solve expensive NP-hard separation problems (which has been the norm in prior research [Kaparis and Letchford, 2010]). Instead, we cheaply generate many candidate cover cuts based on qualitative criteria, lift them, and check for separation only before adding the cut. This approach is effective in experiments with CPLEX.

1.3 Related Work

Cover cuts and their associated separation routines were first shown to be useful in a branch-and-cut framework by Crowder et al. [1983]. Since then, there has been a large body of work studying various computational aspects, both theoretical and practical, of cover cuts, separation routines, and lifting. The seminal work of Gu et al. [2000] showed how sequence-independent lifting can be performed efficiently using g_w for $w(x) = x/\rho_1$. Gu et al. [1998] perform a computational study of sequential lifting, and Wolter [2006] presents some computational results on the interaction between the sequence-independent lifting technique of Gu et al. [2000] and different separation techniques. To our knowledge, this is the only computational study of sequence-independent lifting published to date. Our computational study takes a different approach than prior work. Rather than solving separation problems exactly, which involves expensive optimization, we generate large pools of candidate cover cuts, lift them, and check for separation before adding cuts to the formulation. This approach proves to be effective in our experiments. (The separation problem is NP-hard [Klabjan *et al.*, 1998; Gu *et al.*, 1999], but checking violation is a trivial linear time operation. More on separation can be found in Kaparis and Letchford [2010].) Marchand et al. [2002] and Letchford and Souli [2019; 2020] present other sequence-independent lifting functions based on superadditivity.

[1] Balas [1975] proved that lifting coefficients are sometimes fully determined independent of the lifting order, in which case sequential and sequence-independent lifting are the same and yield a facet. When sequence-independent lifting can be non-trivially performed, ours is the first such result.

2 New Sequence-Independent Lifting Functions: Structure and Properties

We generalize the result of Gu et al. [2000] and also point out an error in their suggested generalization. Gu et al. claim that if $\mu_1 - \lambda \geq \rho_1$, then g_w is superadditive for any nondecreasing $w : [0, \rho_1] \to [0, 1]$ such that $w(x) + w(\rho_1 - x) = 1$. This claim is incorrect (we provide counterexamples in App. A). We show that this claim is correct when restricted to linear w.

Theorem 1. *For $k \in [0, 1/\rho_1]$, let $w_k(x) = kx + \frac{1-k\rho_1}{2}$, and let $g_k = g_{w_k}$. If $\mu_1 - \lambda \geq \rho_1$, g_k is superadditive and undominated.*

The GNS lifting function is given by g_{1/ρ_1}. The proof of Theorem 1 follows the proof that g_{1/ρ_1} is superadditive [Gu *et al.*, 2000] with a few key modifications; we defer it to App. B. Of particular interest is the superadditive lifting function g_0, which we refer to as the *piecewise-constant (PC) lifting function*. The condition $\mu_1 - \lambda \geq \rho_1$ is necessary for superadditivity of g_0 (proven in App. B). The following result shows that the lifting obtained via g_k is dominated by the union of the liftings obtained via g_0 (PC lifting) and g_{1/ρ_1} (GNS lifting). Thus, in order to get as close to $\mathrm{conv}(P)$ as possible, it suffices to study these two lifting functions.

Proposition 1. *Let $k \in (0, 1/\rho_1)$. If $\sum_{j\in C} x_j + \sum_{j\notin C} g_0(a_j)x_j \leq |C| - 1$ and $\sum_{j\in C} x_j + \sum_{j\notin C} g_{1/\rho_1}(a_j)x_j \leq |C| - 1$, then $\sum_{j\in C} x_j + \sum_{j\notin C} g_k(a_j)x_j \leq |C| - 1$.*

Proof. We have $g_k(z) = k\rho_1 g_{1/\rho_1}(z) + (1 - k\rho_1)g_0(z)$ by direct computation, so g_k lifting is a convex combination of GNS and PC lifting. □

Example 1. *Let $C = \{1, 2, 3, 4\}$ and consider a knapsack constraint of the form $16x_1 + 14x_2 + 13x_3 + 9x_4 + \sum_{j\notin C} a_j x_j \leq 44$. C is a minimal cover with $\mu_1 = 16$, $\mu_2 = 30$, $\mu_3 = 43$, $\mu_4 = 52$, $\lambda = 8$, $\rho_1 = 6$, $\rho_2 = 5$, $\rho_3 = 1$, and $\mu_1 - \lambda \geq \rho_1$. Fig. 1 depicts g_0 and g_{1/ρ_1} truncated to the domain $[\mu_1 - \lambda, \mu_3 - \lambda] = [8, 35]$.*

2.1 Comparisons Between PC and GNS Lifting

The following result shows GNS lifting can be arbitrarily worse than PC lifting.

Proposition 2. *For any $\varepsilon > 0, t \in \mathbb{N}$ there exists a knapsack constraint with a minimal cover C of size t such that PC lifting yields $\sum_{j\in C} x_j + \sum_{j\notin C} \frac{1}{2}x_j \leq |C| - 1$ and GNS lifting is dominated by $\sum_{j\in C} x_j + \sum_{j\notin C} \varepsilon x_j \leq |C| - 1$.*

The proof is in Appendix C. At a high level, we construct an instance where the length of S_1, which is ρ_1, is large, and consider coefficients that are at the leftmost part S_1. GNS barely lifts such coefficients, while PC yields lifting coefficients of $1/2$. The next proposition fully characterizes the domination criteria between PC and GNS lifting. Its proof is immediate from the plots in Fig. 1.

Proposition 3. *Assume $\mu_1 - \lambda \geq \rho_1$. Furthermore, suppose $\{j \notin C : \exists h \text{ s.t. } a_j \in S_h\} \neq \emptyset$ (else, GNS and PC trivially produce the same cut). If, for all $j \notin C$,*

1. *$a_j \in S_h \implies \rho_h > \frac{\rho_1}{2}$ and $a_j \leq \mu_h - \lambda + \rho_h - \frac{\rho_1}{2}$ with at least one such $a_j \in S_h$ satisfying $a_j < \mu_h - \lambda + \rho_h - \frac{\rho_1}{2}$, PC strictly dominates GNS.*
2. *$a_j \in S_h \implies \rho_h > \frac{\rho_1}{2}$ and $a_j = \mu_h - \lambda + \rho_h - \frac{\rho_1}{2}$, PC and GNS yield the same cut.*
3. *$a_j \in S_h \implies (\rho_h \leq \frac{\rho_1}{2})$ or $(\rho_h > \frac{\rho_1}{2}$ and $a_j > \mu_h - \lambda + \rho_h - \frac{\rho_1}{2})$, GNS strictly dominates PC.*
4. *Otherwise, neither PC nor GNS dominates the other.*

Example 2. *Consider the constraint $16x_1 + 14x_2 + 13x_3 + 9x_4 + a_5x_5 + a_6x_6 + a_7x_7 \leq 44$ with minimal cover $C = \{1, 2, 3, 4\}$.*

1. *Let $a_5 = 9$, $a_6 = 10$, $a_7 = 23$. GNS yields $x_1 + x_2 + x_3 + x_4 + \frac{1}{6}x_5 + \frac{1}{3}x_6 + \frac{4}{3}x_7 \leq 3$. PC yields the dominant cut $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}x_5 + \frac{1}{2}x_6 + \frac{3}{2}x_7 \leq 3$.*
2. *Let $a_5 = 11$, $a_6 = 17$, $a_7 = 24$. GNS and PC both yield the cut $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}x_5 + x_6 + \frac{3}{2}x_7 \leq 3$.*
3. *Let $a_5 = 12$, $a_6 = 13$, $a_7 = 26$. GNS yields $x_1 + x_2 + x_3 + x_4 + \frac{2}{3}x_5 + \frac{5}{6}x_6 + \frac{11}{6}x_7 \leq 3$. PC yields the weaker cut $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}x_5 + \frac{1}{2}x_6 + \frac{3}{2}x_7 \leq 3$.*
4. *Let $a_5 = 9$, $a_6 = 13$, $a_7 = 24$. GNS yields $x_1 + x_2 + x_3 + x_4 + \frac{1}{6}x_5 + \frac{5}{6}x_6 + \frac{3}{2}x_7 \leq 3$. PC yields $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}x_5 + \frac{1}{2}x_6 + \frac{3}{2}x_7 \leq 3$. Neither cut is dominating.*

Open question: Gu et al. [1999] and Hunsaker and Tovey [2005] construct knapsack problems where branch-and-cut requires a tree of exponential size, even when *all lifted cover inequalities* (all inequalities of the form (1)) are added to the formulation. Do there exist knapsack problems where branch-and-cut requires exponential-size trees when all GNS-lifted cover inequalities are added, but does not when PC-lifted cover inequalities are added (or vice versa)?

2.2 Facet Defining Inequalities From Sequence-Independent Lifting

We now provide a broad set of sufficient conditions under which PC lifting yields facet-defining inequalities. Our result relies on the following complete characterization of the facets of the knapsack polytope obtained from lifting minimal cover cuts, due to Balas and Zemel [1978; 2020], which we restate using the notation and terminology of Gu et al. [2000] and Conforti et al. [2014, sec. 7.2]. Given a minimal cover C and $j \notin C$, let $h(j)$ be the index such that $\mu_{h(j)} \leq a_j < \mu_{h(j)+1}$.

Theorem 2 (Balas and Zemel [1978])**.** *Let C be a minimal cover. Let $J = \{j \notin C : a_j > \mu_{h(j)+1} - \lambda\}$ and let $I = (\{1, \ldots, n\} \setminus C) \setminus J$. Let $\mathcal{Q}(J) = \{Q \subseteq J : \sum_{j\in Q} a_j \leq b, Q \neq \emptyset\}$. The inequality $\sum_{j\in C} x_j + \sum_{j\notin C} \alpha_j x_j \leq |C| - 1$ is a facet of $\mathrm{conv}(P)$ if and only if $\alpha_j = h(j) + \delta_j \cdot \mathbf{1}(j \in J)$ where each $\delta_j \in [0, 1]$ and $\boldsymbol{\delta} = (\delta_j)_{j\in J}$ is a vertex of the polyhedron $T =$*

$$\left\{ \boldsymbol{\delta} \in \mathbb{R}^{|J|} : \sum_{j\in Q} \delta_j \leq f\Big(\sum_{j\in Q} a_j\Big) - \sum_{j\in Q} h(j) \,\forall\, Q \in \mathcal{Q}(J) \right\}.$$

The characterization of facets based on T in Theorem 2 does not translate to a tractable method of deriving facet-defining inequalities, since one would need to enumerate the

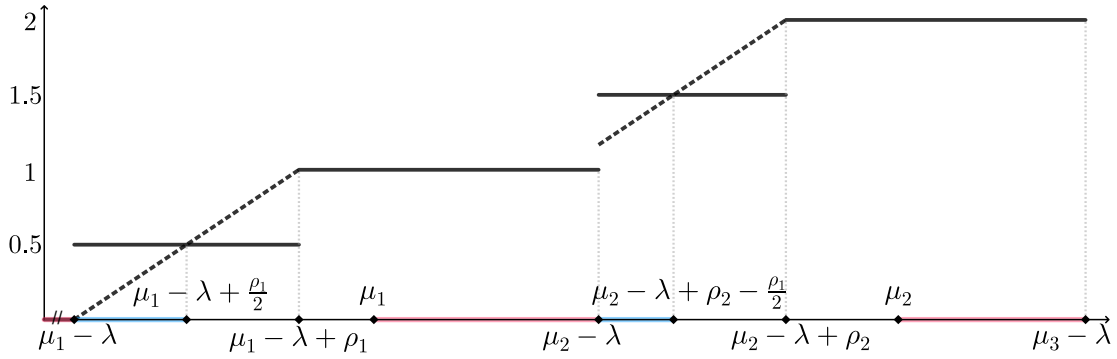

Figure 1: The PC lifting function g_0 is the piecewise constant step function depicted by the solid black lines. The GNS lifting function g_{1/ρ_1} is obtained by replacing the solid lines in the intervals S_h with the depicted dashed lines. If all coefficients of variables being lifted lie in the blue and red regions with at least three coefficients in the leftmost blue region, PC lifting is facet-defining and dominates GNS lifting (Theorem 3).

vertices of T. In fact, Hartvigsen and Zemel [1992] show the question of determining whether or not a cutting plane is a valid lifted cover cut is co-NP-complete. Deciding whether or not a cutting plane is a facet-defining lifted cover cut is in D^{P} (D^{P} is a complexity class introduced by Papadimitriou and Yannakakis [1982] to characterize the complexity of facet recognition). Critically, these complexity results hold when the underlying cover is given as input.

Our result, to the best of our knowledge, provides *the first broad set of sufficient conditions under which sequence-independent liftings that can be efficiently computed from the underlying minimal cover—via PC lifting—are facet defining.* We now state and prove our result.

Theorem 3. *Let $C = \{1, \dots, t\}$, $a_1 \geq \cdots \geq a_t$, be a minimal cover such that $\mu_1 - \lambda \geq \rho_1 > 0$. Suppose $|\{j \notin C : a_j \in S_1\}| \geq 3$ and for all $j \notin C$:*

$$a_j \in S_h \implies \rho_h > \frac{\rho_1}{2} \text{ and } a_j \leq \mu_h - \lambda + \rho_h - \frac{\rho_1}{2},$$
$$a_j \in F_h \implies a_j \geq \mu_h \text{ (equivalently } h(j) \geq h).$$

Then, the cut

$$\sum_{j \in C} x_j + \sum_{j \notin C} g_0(a_j) x_j \leq |C| - 1,$$

obtained via PC lifting, defines a facet of $\mathrm{conv}(P)$.

Proof. First we show that $J = \cup_{h \geq 1}\{j \notin C : a_j \in S_h\}$ and $I = \cup_{h \geq 0}\{j \notin C : a_j \in F_h\}$. Let $j \notin C$ be such that $a_j \in S_h$. We have $a_j > \mu_h - \lambda > \mu_{h-1}$ (as $\lambda \leq a_i$ for any $i \in C$) and $a_j \leq \mu_h - \lambda + \rho_h - \frac{\rho_1}{2} < \mu_h - \lambda + \rho_h < \mu_h$ (the final inequality follows directly from expanding out μ_h and ρ_h). So $h(j) = h - 1$, and as $a_j > \mu_h - \lambda = \mu_{h(j)+1} - \lambda$, $j \in J$. Next, let $j \notin C$ be such that $a_j \in F_h$. Then, $a_j \leq \mu_{h+1} - \lambda < \mu_{h+1}$, and by assumption $a_j \geq \mu_h$, so $h(j) = h$. Therefore $a_j \leq \mu_{h(j)+1} - \lambda$, and so $j \in I$. The facts that $a_j \in S_h \implies h(j) = h - 1$ and $a_j \in F_h \implies h(j) = h$ will be used repeatedly in the remainder of the proof.

We now determine the constraints defining the polyhedron $T \subset \mathbb{R}^{|J|}$ in Theorem 2. (For $j \in I$, PC lifting produces a coefficient of $h(j)$, as required by Theorem 2.) Partition J into two sets $J = J_1 \cup J_{>1}$ where $J_1 = \{j \notin C : a_j \in S_1\}$ and $J_{>1} = \cup_{h>1}\{j \notin C : a_j \in S_h\}$. Each singleton $Q = \{j\} \in \mathcal{Q}(J)$ yields the constraint $\delta_j \leq 1$. Consider now $Q = \{j_1, j_2\} \in \mathcal{Q}(J)$. We consider two cases, one where $j_1 \in J_1$ and the other where $j_1 \in J_{>1}$. First, let $j_1 \in J_1$. Let h be such that $a_{j_2} \in S_h$. We have $a_{j_1} + a_{j_2} > a_{j_2} > \mu_h - \lambda$, so $f(a_{j_1} + a_{j_2}) \geq h$, and $a_{j_1} + a_{j_2} \leq \mu_1 - \lambda + \frac{\rho_1}{2} + \mu_h - \lambda + \rho_h - \frac{\rho_1}{2} = a_1 - \lambda + \mu_h - \lambda + (a_{h+1} - a_1 + \lambda) = \mu_h + a_{h+1} - \lambda = \mu_{h+1} - \lambda$, so $f(a_{j_1} + a_{j_2}) \leq h$. Therefore $f(a_{j_1} + a_{j_2}) = h$, and we get the constraint $\delta_{j_1} + \delta_{j_2} \leq f(a_{j_1} + a_{j_2}) - h(j_1) - h(j_2) = h - 0 - (h - 1) = 1$. Suppose now that $j_1, j_2 \in J_{>1}$ (if $j_2 \in J_1$ that is handled by the first case). Let h_1, h_2 be the integers such that $a_{j_1} \in S_{h_1}$ and $a_{j_2} \in S_{h_2}$. We have $a_{j_1} + a_{j_2} > \mu_{h_1} - \lambda + \mu_{h_2} - \lambda = (a_1 + \cdots + a_{h_1}) + (a_1 + \cdots + a_{h_2}) - 2\lambda > (a_1 + \cdots + a_{h_1 - 1}) + (a_1 + \cdots + a_{h_2}) - \lambda > \mu_{h_1 + h_2 - 1} - \lambda$ so $f(a_{j_1} + a_{j_2}) \geq h_1 + h_2 - 1$, and $f(a_{j_1} + a_{j_2}) - h(j_1) - h(j_2) \geq 1$. So for such pairs, we obtain a constraint $\delta_{j_1} + \delta_{j_2} \leq H(j_1, j_2)$, where $H(j_1, j_2) \geq 1$. Finally, we consider $Q \in \mathcal{Q}(J)$ with $|Q| \geq 3$. For $j \in Q$ let h_j be the integer such that $a_j \in S_{h_j}$. We claim that

$$\sum_{j \in Q} a_j > \mu_{\sum_{j \in Q} h_j - \lfloor |Q|/2 \rfloor} - \lambda.$$

This claim is succinctly proven using the quantities used to prove Theorem 1 (defined by Gu et al. [2000] to prove superadditivity of g_{1/ρ_1}). To avoid notational clutter, we defer its proof to App. D. The claim implies $f(\sum_{j \in Q} a_j) \geq \sum_{j \in Q} h_j - \lfloor |Q|/2 \rfloor$, so the constraint induced by Q is of the form $\sum_{j \in Q} \delta_j \leq H(Q)$, where

$$\begin{aligned} H(Q) &:= f\Big(\sum_{j \in Q} a_j\Big) - \sum_{j \in Q} h(j) \\ &\geq \sum_{j \in Q} h_j - \lfloor |Q|/2 \rfloor - \sum_{j \in Q} (h_j - 1) \\ &= \lceil |Q|/2 \rceil. \end{aligned}$$

We can now write down a complete description of T as

$$\left\{ \boldsymbol{\delta} \in \mathbb{R}^{|J|} : \begin{array}{l} (1)\, \delta_j \leq 1 \,\forall\, j \in J \\ (2)\, \delta_i + \delta_j \leq 1 \,\forall\, (i,j) \in J_1 \times J \\ (3)\, \delta_i + \delta_j \leq H(i,j) \,\forall\, (i,j) \in J_{>1} \times J_{>1} \\ (4)\, \sum_{j \in Q} \delta_j \leq H(Q) \,\forall\, Q \in \mathcal{Q}(J), |Q| \geq 3 \end{array} \right\}$$

where $H(i,j) \geq 1$ for all $(i,j) \in J_{>1} \times J_{>1}$ and $H(Q) \geq \lceil |Q|/2 \rceil$ for all $Q \in \mathcal{Q}(J)$, $|Q| \geq 3$. We argue that $\boldsymbol{\delta} = (1/2, \ldots, 1/2)$ is a vertex of T. Constraints of type (1), (3), and (4) are satisfied, and type (2) constraints are tight. Fix distinct $j_1, j_2, j_3 \in J_1$. The set of $|J|$ type (2) constraints $\{\delta_j + \delta_{j_1} \leq 1 \; \forall \; j \in J \setminus \{j_1\}\} \cup \{\delta_{j_2} + \delta_{j_3} \leq 1\}$ is linearly independent, and hence $\boldsymbol{\delta} = (1/2, \ldots, 1/2)$ is a vertex of T. PC lifting produces precisely the coefficients prescribed by Theorem 2: $g_0(a_j) = h(j)$ for $j \in I$ and $g_0(a_j) = h(j) + \frac{1}{2}$ for $j \in J$, so we are done. □

Fig. 1 illustrates the sufficient conditions of Theorem 3. While a facet-defining PC lifting can be efficiently obtained given a minimal cover satisfying the sufficient conditions of Theorem 3, we do not show how to find a minimal cover satisfying these conditions. We conjecture that finding such a cover is NP-hard.

Example 3. *Consider the constraint* $16x_1 + 14x_2 + 13x_3 + 9x_4 + 9x_5 + 10x_6 + 11x_7 + 23x_8 \leq 44$ *with minimal cover* $C = \{1,2,3,4\}$. *GNS lifting yields the cut* $x_1 + x_2 + x_3 + x_4 + \frac{1}{6}x_5 + \frac{1}{3}x_6 + \frac{1}{2}x_7 + \frac{4}{3}x_8 \leq 3$ *and PC lifting yields the strictly dominant facet-defining cut* $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6 + x_7) + \frac{3}{2}x_8 \leq 3$.

Other Half-Integral Liftings. There exist facet-defining lifted cover inequalities with half-integral coefficients that cannot be obtained via PC lifting. Balas and Zemel [1978] provide an example of a knapsack constraint and minimal cover for which $\boldsymbol{\delta} = (1/2, \ldots, 1/2)$ is a vertex of T, but $\mu_1 - \lambda \geq \rho_1$ *does not hold*. It is an interesting open question to investigate when such facets arise and how to (efficiently) find them. The lifting procedure of Letchford and Souli [2019] also produces half-integral coefficients, but it is unclear when it can yield facets (it produces cuts that in general are incomparable with ours as their lifting function is also undominated). In Example 3, Letchford and Souli's lifting technique (which is implemented in version 9.5 of the FICO Xpress solver [Perregard, 2024]) yields the cut $x_1 + x_2 + x_3 + x_4 + x_8 \leq 3$, a significantly weaker cut than the facet-defining PC cut. The numerical properties of half-integral cuts make them desirable for implementation within a solver, and the computational overhead of PC lifting (sorting the cover elements) is the same as Letchford and Souli's lifting.

It would be interesting to derive similar conditions under which GNS lifting defines facets. The following sufficient conditions are immediate, but it is likely that a stronger result could be derived. We leave this as an open question.

Proposition 4. *If for all* $j \notin C$, $a_j \in S_h \implies a_j = \mu_h - \lambda + \rho_h$, *GNS lifting strictly dominates PC lifting and defines a facet of* $\operatorname{conv}(P)$.

Proof. The condition implies g_{1/ρ_1} coincides with the lifting function f on all a_j, $j \notin C$, which (*e.g.*, Prop. 7.2 in Conforti et al. [2014]) means GNS lifting is facet defining. □

3 Experimental Evaluation

We evaluate our new sequence-independent lifting techniques in conjunction with a number of methods for generating the minimal cover cuts that are to be lifted. We describe each component of the experimental setup below.

Cover Cut Generation. Let $\sum_{j=1}^{n} a_j x_j \leq b$ be a knapsack constraint, let $\boldsymbol{x}^{\mathsf{LP}}$ be the LP optimal solution at a given node of the branch-and-cut tree, and let $\mathcal{I} = \{i : x_i^{\mathsf{LP}} > 0\}$. Enumerate $\mathcal{I}$ as $\mathcal{I} = \{1, \ldots, s\}$ (relabeling variables if necessary). We do not include variables not in $\mathcal{I}$ in any of the minimal covers, since these do not contribute to the violation of $\boldsymbol{x}^{\mathsf{LP}}$ (though such variables may be assigned a nonzero coefficient in the lifted cover cut). Next, we present the five cover cut generation techniques that we use in experiments.

Contiguous covers. First, sort the variables in $\mathcal{I}$ in descending order of weight; without loss of generality relabel them so that $a_1 \geq a_2 \geq \cdots \geq a_s$. For each $i \in \{1, \ldots, s\}$, let $j \in \{i+1, \ldots, s\}$ be such that $C = \{i, i+1, \ldots, j\}$ is a minimal cover (if such a j exists). This is the *contiguous* cover starting at i. We generate all such contiguous cover cuts for each i. (Balcan et al. [2022a] introduced these cover cuts, though they did not lift them nor did they restrict to $x_i^{\mathsf{LP}} > 0$.)

Spread covers. As before, sort the variables in $\mathcal{I}$ in descending order of weight; $a_1 \geq \cdots \geq a_s$. For each i, let $j \in \{i+1, \ldots, s\}$ be *maximal* (if such a j exists) such that there exists $k \in \{j+1, \ldots, s\}$ such that $C = \{i\} \cup \{j, \ldots, k\}$ is a minimal cover. Intuitively, i can be thought of as a heavy "head", and the "tail" from j to k is as light as possible. We coin this the *spread* cover with head i. We generate all such spread cover cuts for each i.

Heaviest contiguous cover. We define this as the contiguous cover starting at 1.

Default cover. Sort (and relabel) the variables in $\mathcal{I}$ in descending order of LP value so that $x_1^{\mathsf{LP}} \geq \cdots \geq x_s^{\mathsf{LP}}$. Let j be minimal so that $\{1, \ldots, j\}$ is a cover. Then, evict items, lightest first, until the cover is minimal. We coin this the *default* cover. These cover cuts are loosely based on the "default" separation routine implemented by Gu et al. [1998]. (Their routine was for sequential lifting and does not directly carry over to our setting.) Wolter [2006] tested similar routines.

Bang-for-buck cover. Sort (and relabel) the variables in $\mathcal{I}$ in descending order of "bang-for-buck" so that $\frac{c_1}{a_1} \geq \cdots \frac{c_s}{a_s}$, where c_i is variable i's objective coefficient. Let j be minimal so that $\{1, \ldots, j\}$ is a cover. Then, evict items, lightest first, until the cover is minimal. We coin this the *bang-for-buck* cover.

Example 4. *Consider the knapsack constraint* $10x_1 + 9x_2 + 8x_3 + 7x_4 + 6x_5 + 6x_6 + 5x_7 + 4x_8 \leq 26$, *suppose the LP optimal solution at the current branch-and-cut node is* $\boldsymbol{x}^{\mathsf{LP}} = (0.1, 0.8, 0.7, 0.4, 0, 1, 0.2, 0.8)$, *and suppose* $\boldsymbol{c} = (5, 7, 9, 1, 2, 6, 6, 5)$. *We have* $\mathcal{I} = \{1, 2, 3, 4, 6, 7, 8\}$. *The contiguous minimal covers are* $\{1, 2, 3\}$, $\{2, 3, 4, 6\}$, $\{3, 4, 6, 7, 8\}$. *The spread minimal covers are* $\{1, 4, 6, 7\}$, $\{2, 4, 6, 7\}$, $\{3, 4, 6, 7, 8\}$. *The heaviest contiguous cover is* $\{1, 2, 3\}$. *The default cover is* $\{2, 3, 6, 8\}$. *The bang-for-buck cover is* $\{2, 3, 6, 7\}$.

Lifting. We evaluate three lifting methods. (1) *GNS:* The cover cut is lifted using g_{1/ρ_1}. (2) *PC:* If $\mu_1 - \lambda \geq \rho_1$ the cover cut is lifted using g_0, and otherwise it is lifted using g_{1/ρ_1}. (3) *Smart:* If $\mu_1 - \lambda \geq \rho_1$, two liftings are generated: one with g_0 and one with g_{1/ρ_1}. If either lifting dominates

Algorithm 1 Lifted cover cut generation at a node of branch-and-cut

Input: IP data $\boldsymbol{c}, \boldsymbol{A}, \boldsymbol{b}$, LP optimum at current node $\boldsymbol{x}^{\mathsf{LP}}$, per-node cut limit ℓ

Initialize $\mathsf{cuts} = \emptyset$.
for each knapsack constraint $\boldsymbol{a}^\top \boldsymbol{x} \leq b$ **do**
  for each $\mathsf{cut} \in \texttt{CoverCuts}(\boldsymbol{a}, b, \boldsymbol{c}, \boldsymbol{x}^{\mathsf{LP}})$ **do**
    for each $\mathsf{liftedcut} \in \texttt{Lift}(\mathsf{cut})$ **do**
      if liftedcut separates $\boldsymbol{x}^{\mathsf{LP}}$ **then**
        $\mathsf{cuts} \leftarrow \mathsf{cuts} \cup \{\mathsf{liftedcut}\}$.
Add the top $\min\{\ell, |\mathsf{cuts}|\}$ cuts in cuts with respect to efficacy.

the other, the weaker lifting is discarded. If $\mu_1 - \lambda < \rho_1$, the cover cut is only lifted using g_{1/ρ_1}.

Integration with Branch-and-Cut. Alg. 1 is the pseudocode for adding lifted cover cuts at a node of the branch-and-cut tree and is called once at every node of the tree. It uses the prescribed lifting method `Lift` atop the prescribed cover cut generation method `CoverCuts` for each constraint, and adds the resulting lifted cut to a set of candidate cuts if it separates the current LP optimum $\boldsymbol{x}^{\mathsf{LP}}$. It adds the ℓ deepest cuts among the candidate cuts to the formulation. (The depth or *efficacy* of a cut $\boldsymbol{\alpha}^\top \boldsymbol{x} \leq \beta$ is the quantity $\frac{\boldsymbol{\alpha}^\top \boldsymbol{x}^{\mathsf{LP}} - \beta}{\|\boldsymbol{\alpha}_2\|}$ and measures the distance between the cut and $\boldsymbol{x}^{\mathsf{LP}}$.) The ℓ cuts are added in a single round at the current node, and no further cuts are generated at that node. Alg. 1 does not solve NP-hard separation problems and instead relies on fast ways of generating candidate cuts through `CoverCuts`, and only adds those that separate $\boldsymbol{x}^{\mathsf{LP}}$.

Experimental Results. We evaluated our techniques on five problem distributions: two distributions over winner-determination problems in multi-unit combinatorial auctions (*decay-decay* [Sandholm *et al.*, 2002] and *multipaths* from CATS version 1.0 [Leyton-Brown *et al.*, 2000]) and three distributions over multiple knapsack problems (*uncorrelated* and *weakly correlated* benchmark distributions from Fukunaga [2011] and *Chvátal* from Balcan et al. [2022a]). We ran experiments in C++ using the callable library of CPLEX version 20.1 on a 64-core machine, and implemented Alg. 1 with $\ell = 10$ within a cut callback. We generated 100 integer programs from each distribution. We set a 1 hour time limit, allocated 16GB of RAM, and used one thread for each integer program.

We present illustrative results in Fig. 2, focusing on settings where PC lifting had significant impact. The full set of experiments are in App. E. We focus on tree size as the performance measure, but we provide full run-time results in App. E. Each curve is labeled `setting/lifting/covers` or `setting/CPLEX`. The `setting` label is either empty, `np`, or `d`. Empty corresponds to all cuts, all heuristics, and presolve off. A setting of `np` denotes presolve off and all other settings untouched. A setting of `d` denotes the default settings (this setting is incompatible with our lifting implementations since presolve is incompatible with cut callbacks). If an underscore follows the label, CPLEX's internal cover cut generation is turned off; otherwise it is on. To ensure fair comparisons, in all CPLEX runs not involving our new techniques (those labeled `setting/CPLEX`), we register a dummy cut callback that does nothing. This disables proprietary techniques such as "dynamic search" which do not support callbacks and thus do not support experiments of the type required in this paper.

We now describe the performance plots (Fig. 2), which display how many instances each method was able to solve using trees smaller than the prescribed size on the x-axis within the 1 hour time limit. *Weakly correlated, contiguous covers (top left):* GNS solves 81 instances, Smart solves 83 instances, and PC solves 88 instances. These all outperform default CPLEX which solves 80 instances and builds trees an order of magnitude larger. *Uncorrelated, contiguous covers (top right):* GNS solves 43 instances, Smart solves 48 instances, and PC solves 55 instances, while default CPLEX solves 78 instances. *Chvátal, heaviest covers (middle left):* PC lifting dramatically outperforms the other methods. Here, all CPLEX parameters are turned off, so we are directly comparing our lifted cover cut implementations with CPLEX's own cover cut generation routines. PC and Smart strictly outperform GNS and CPLEX (GNS is the only one unable to solve all 100 problems), and the largest tree size required by PC is an order of magnitude smaller than any of the other methods. This translates into a run-time improvement as well due to PC lifting (Fig. 6 in the appendix). On the auction instances, there is little discernible performance difference between the lifting methods. *Decay-decay, spread covers (middle right):* our lifting implementations with all other CPLEX parameters off dramatically outperform default CPLEX on problems requiring trees of size $> 10^4$ (though default CPLEX solves all 100 problems while our methods do not.) *Multipaths, spread covers (bottom left):* we once again directly compare our lifted cover cut implementations with CPLEX's internal cover cut generation. Here, spread covers yield over an order of magnitude reduction in tree size relative to CPLEX, while default covers perform extremely poorly. We observed that contiguous and spread covers generally resulted in the best performance (with heaviest contiguous covers on par), and default and bang-for-buck covers performed much worse.

While our techniques often led to significantly smaller trees than CPLEX, this did not translate to significant run-time improvements. However, in most settings our techniques were not too much slower than CPLEX and sometimes they were faster. Full run-time performance plots are in App. E. A possibility for this is that we did not limit the total number of cuts added throughout the tree, causing the formulation to grow very large. We ran an experiment to investigate the run-time impact of varying the number of cuts allowed (i) at each node (ℓ in Alg. 1) and (ii) throughout the entire search tree. We plot the average run-time (time limit of 1 hour) over the first 10 weakly-correlated instances using `pc/contiguous`, visualized as a heatmap (Fig. 2; bottom right). The best settings (limiting the overall number of cuts to 800) yield an average run-time of around 300 seconds and solved all 10 instances. Default CPLEX (`d/CPLEX`) averaged roughly 900 seconds and only solved 9 instances to optimality.

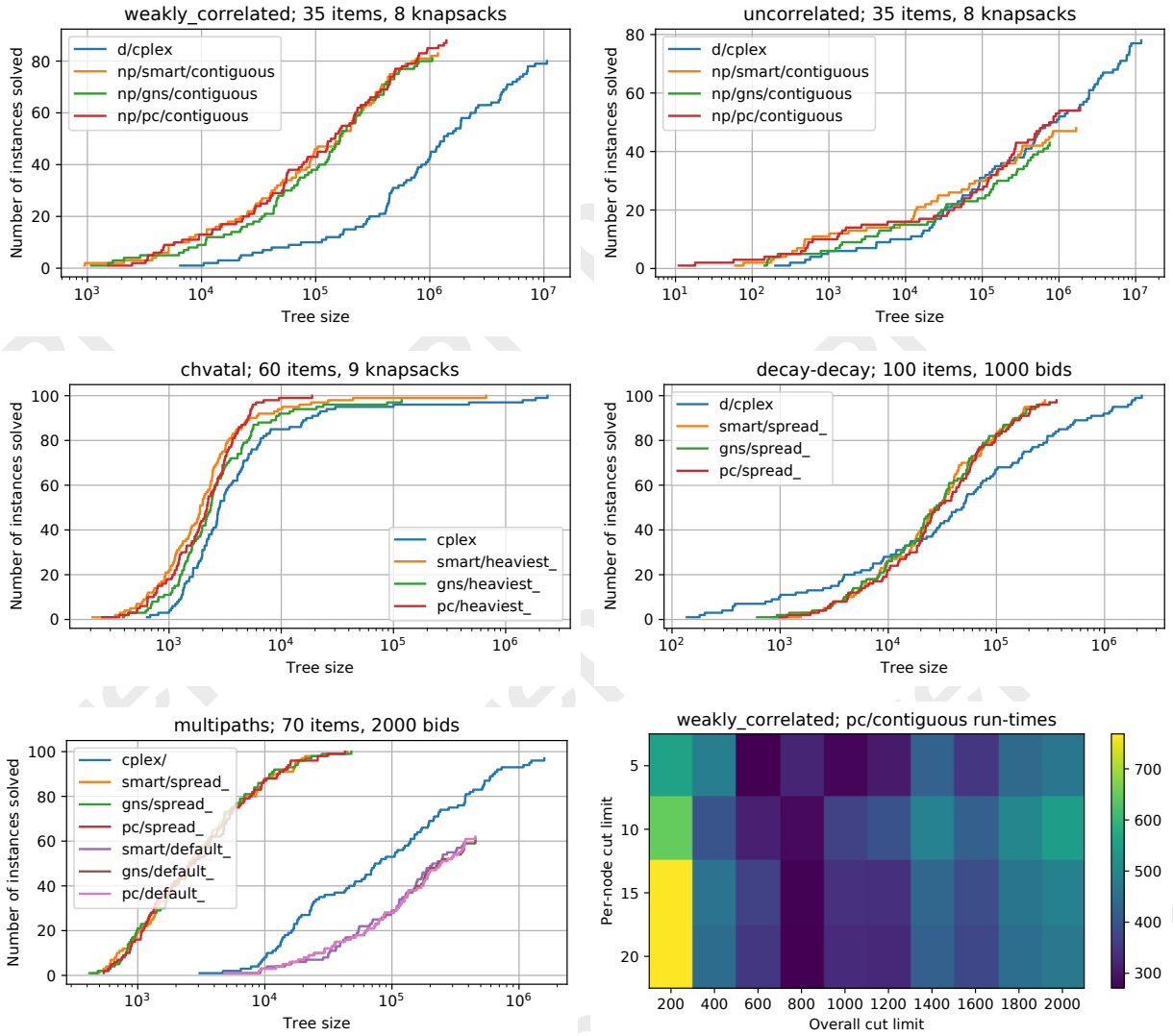

Figure 2: Illustrative experiments comparing different lifting methods. The first five plots are performance plots for the five different problem distributions, with various parameter settings. The heatmap illustrates the effect of varying the per-node cut limit and overall cut limit on run-time (avg. over first 10 weakly-correlated instances).

4 Conclusions and Future Research

In this paper we showed that PC lifting can be a useful alternative to GNS lifting. We proved that under some sufficient conditions, PC lifting is facet-defining. To our knowledge, these are the first broad conditions for facet-defining sequence-independent liftings that can be efficiently computed from the underlying cover. We invented new cover cut generation routines, which in conjunction with our new lifting techniques, displayed promising practical performance.

There are a number of important future research directions that stem from our findings. First, a much more extensive experimental evaluation of PC lifting is needed. We have made several simplifying design choices, including (i) adding only one wave of lifted cover cuts at each node; (ii) ranking cuts solely based on efficacy (efficacy is not always the best quality to prioritize [Balcan *et al.*, 2022a; Balcan *et al.*, 2022b; Turner *et al.*, 2023]); (iii) keeping ℓ constant across nodes while it could be a tuned hyperparameter, or even adjusted dynamically during search. Our experiments show promise even with these fixed choices, but a more thorough suite of tests could find how to best exploit the potential of our new techniques. Another direction here is to use machine learning (which has already been used to tune cutting plane selection policies [Balcan *et al.*, 2021; Li *et al.*, 2023]) to decide when to use PC or GNS lifting. There might also be additional ways of determining what lifting method to use based on problem structure, and detecting that automatically. Finally, PC lifting possesses strong numerical properties since it always involves half-integral coefficients. That, along with its ability to yield facets, makes PC lifting practically relevant for solvers.

Acknowledgments

This material is based on work supported by the Vannevar Bush Faculty Fellowship ONR N00014-23-1-2876, NSF grants RI-2312342 and RI-1901403, ARO award W911NF2210266, NIH award A240108S001 and Defense Advanced Research Projects Agency under cooperative agreement HR00112020003.

References

[Balas and Zemel, 1978] Egon Balas and Eitan Zemel. Facets of the knapsack polytope from minimal covers. *SIAM Journal on Applied Mathematics*, 34(1):119–148, 1978.

[Balas, 1975] Egon Balas. Facets of the knapsack polytope. *Mathematical programming*, 8(1):146–164, 1975.

[Balcan *et al.*, 2021] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. Sample complexity of tree search configuration: Cutting planes and beyond. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[Balcan *et al.*, 2022a] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. Improved sample complexity bounds for branch-and-cut. In *28th International Conference on Principles and Practice of Constraint Programming (CP)*, 2022.

[Balcan *et al.*, 2022b] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. Structural analysis of branch-and-cut and the learnability of Gomory mixed integer cuts. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[Conforti *et al.*, 2014] Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, et al. *Integer programming*, volume 271. Springer, 2014.

[Crowder *et al.*, 1983] Harlan Crowder, Ellis L Johnson, and Manfred Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 1983.

[Fukunaga, 2011] Alex S Fukunaga. A branch-and-bound algorithm for hard multiple knapsack problems. *Annals of Operations Research*, 184(1):97–119, 2011.

[Gu *et al.*, 1998] Zonghao Gu, George L Nemhauser, and Martin WP Savelsbergh. Lifted cover inequalities for 0-1 integer programs: Computation. *INFORMS Journal on Computing*, 1998.

[Gu *et al.*, 1999] Zonghao Gu, George L Nemhauser, and Martin WP Savelsbergh. Lifted cover inequalities for 0-1 integer programs: Complexity. *INFORMS Journal on Computing*, 1999.

[Gu *et al.*, 2000] Zonghao Gu, George L Nemhauser, and Martin WP Savelsbergh. Sequence independent lifting in mixed integer programming. *Journal of Combinatorial Optimization*, 4(1):109–129, 2000.

[Hartvigsen and Zemel, 1992] David Hartvigsen and Eitan Zemel. The complexity of lifted inequalities for the knapsack problem. *Discrete Applied Mathematics*, 39(2):113–123, 1992.

[Hojny *et al.*, 2020] Christopher Hojny, Tristan Gally, Oliver Habeck, Hendrik Lüthen, Frederic Matter, Marc E Pfetsch, and Andreas Schmitt. Knapsack polytopes: a survey. *Annals of Operations Research*, 292:469–517, 2020.

[Hunsaker and Tovey, 2005] Brady Hunsaker and Craig A Tovey. Simple lifted cover inequalities and hard knapsack problems. *Discrete Optimization*, 2(3):219–228, 2005.

[Kaparis and Letchford, 2010] Konstantinos Kaparis and Adam N Letchford. Separation algorithms for 0-1 knapsack polytopes. *Mathematical programming*, 124:69–91, 2010.

[Klabjan *et al.*, 1998] Diego Klabjan, George L Nemhauser, and Craig Tovey. The complexity of cover inequality separation. *Operations Research Letters*, 23(1-2):35–40, 1998.

[Letchford and Souli, 2019] Adam N Letchford and Georgia Souli. On lifted cover inequalities: A new lifting procedure with unusual properties. *Operations Research Letters*, 47(2):83–87, 2019.

[Letchford and Souli, 2020] Adam N Letchford and Georgia Souli. Lifting the knapsack cover inequalities for the knapsack polytope. *Operations Research Letters*, 48(5):607–611, 2020.

[Leyton-Brown *et al.*, 2000] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the 2nd ACM conference on Electronic Commerce (EC)*, pages 66–76, 2000.

[Li *et al.*, 2023] Sirui Li, Wenbin Ouyang, Max B. Paulus, and Cathy Wu. Learning to configure separators in branch-and-cut. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[Marchand *et al.*, 2002] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 2002.

[Papadimitriou and Yannakakis, 1982] Christos H Papadimitriou and Mihalis Yannakakis. The complexity of facets (and some facets of complexity). In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 255–260, 1982.

[Perregard, 2024] Michael Perregard. New in FICO Xpress solver. https://plato.asu.edu/ftp/informs_talks_2024/perregard.pdf, 2024.

[Sandholm *et al.*, 2002] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 69–76, 2002.

[Turner *et al.*, 2023] Mark Turner, Timo Berthold, Mathieu Besançon, and Thorsten Koch. Cutting plane selection with analytic centers and multiregression. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pages 52–68. Springer, 2023.

[Wolsey, 1977] Laurence A Wolsey. Valid inequalities and superadditivity for 0–1 integer programs. *Mathematics of Operations Research*, 2(1):66–77, 1977.

[Wolter, 2006] Kati Wolter. Implementation of cutting plane separators for mixed integer programs. *Dipolma thesis, Technische Universität Berlin*, 2006.