A fast combinatorial algorithm for the bilevel knapsack problem with interdiction constraints

Noah Weninger

Joint work with Ricardo Fukasawa

IPCO 2023 @ Madison
June 22, 2023

UNIVERSITY OF WATERLOO
FACULTY OF MATHEMATICS
Department of Combinatorics and Optimization

Bilevel knapsack with interdiction constraints (BKP)

- A generalization of 0-1 knapsack where an adversary can block access to (interdict) some items

Bilevel knapsack with interdiction constraints (BKP)

- A generalization of 0-1 knapsack where an adversary can block access to (interdict) some items
- We are interested in solving this problem exactly

Bilevel knapsack with interdiction constraints (BKP)

- A generalization of 0-1 knapsack where an adversary can block access to (interdict) some items
- We are interested in solving this problem exactly
- Objective: win the horse race

Game theoretic interpretation of BKP

Given: n items, weights $w^U, w^L \in \mathbb{Z}^n_{\geq 0}$, profits $p \in \mathbb{Z}^n_{\geq 0}$, capacities $C^U, C^L \in \mathbb{Z}_{\geq 0}$.

- **Round 1:** Leader (adversary) selects $X \subseteq [n]$ s.t. $\sum_{i \in X} w_i^U \leq C^U$

Game theoretic interpretation of BKP

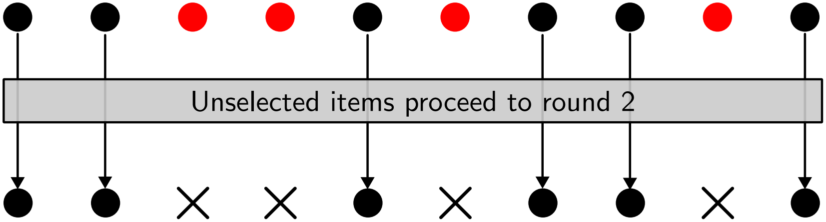
Given: n items, weights $w^U, w^L \in \mathbb{Z}^n_{\geq 0}$, profits $p \in \mathbb{Z}^n_{\geq 0}$, capacities $C^U, C^L \in \mathbb{Z}_{\geq 0}$.

- **Round 1:** Leader (adversary) selects $X \subseteq [n]$ s.t. $\sum_{i \in X} w_i^U \leq C^U$

Game theoretic interpretation of BKP

Given: n items, weights $w^U, w^L \in \mathbb{Z}^n_{\geq 0}$, profits $p \in \mathbb{Z}^n_{\geq 0}$, capacities $C^U, C^L \in \mathbb{Z}_{\geq 0}$.

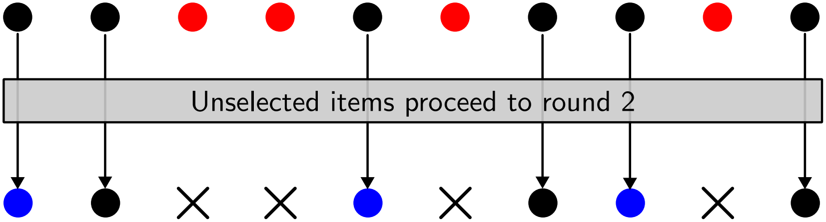
- **Round 1:** Leader (adversary) selects $X \subseteq [n]$ s.t. $\sum_{i \in X} w_i^U \leq C^U$



Game theoretic interpretation of BKP

Given: n items, weights $w^U, w^L \in \mathbb{Z}^n_{\geq 0}$, profits $p \in \mathbb{Z}^n_{\geq 0}$, capacities $C^U, C^L \in \mathbb{Z}_{\geq 0}$.

- **Round 1:** Leader (adversary) selects $X \subseteq [n]$ s.t. $\sum_{i \in X} w_i^U \leq C^U$

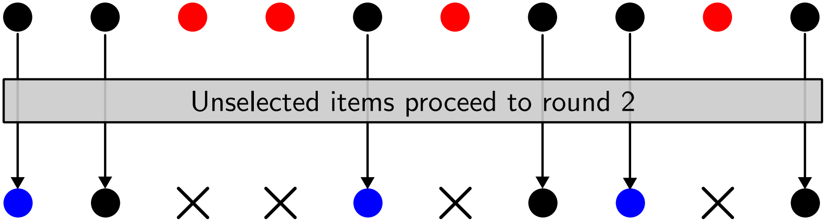


- **Round 2:** Follower selects $Y \subseteq [n] \setminus X$ s.t. $\sum_{i \in Y} w_i^L \leq C^L$ and $\sum_{i \in Y} p_i$ is maximized

Game theoretic interpretation of BKP

Given: n items, weights $w^U, w^L \in \mathbb{Z}^n_{\geq 0}$, profits $p \in \mathbb{Z}^n_{\geq 0}$, capacities $C^U, C^L \in \mathbb{Z}_{\geq 0}$.

- **Round 1:** Leader (adversary) selects $X \subseteq [n]$ s.t. $\sum_{i \in X} w_i^U \leq C^U$



- **Round 2:** Follower selects $Y \subseteq [n] \setminus X$ s.t. $\sum_{i \in Y} w_i^L \leq C^L$ and $\sum_{i \in Y} p_i$ is maximized
- **Objective**: Select X to minimize the maximum profit the Follower can get

Bilevel programming interpretation of BKP

Given: n items, weights $w^U, w^L \in \mathbb{Z}^n_{\geq 0}$, profits $p \in \mathbb{Z}^n_{\geq 0}$, capacities $C^U, C^L \in \mathbb{Z}_{\geq 0}$.

Objective: $\min_{X \in \mathcal{U}} \max_{Y \in \mathcal{L}(X)} \sum_{i \in Y} p_i$

where

$$\mathcal{U} = \left\{ X \subseteq \{1, \ldots, n\} : \sum_{i \in X} w_i^U \leq C^U \right\} \qquad \text{(upper/leaders knapsack)}$$

$$\mathcal{L}(X) = \left\{ Y \subseteq \{1, \ldots, n\} \setminus X : \sum_{i \in Y} w_i^L \leq C^L \right\} \qquad \text{(lower/followers knapsack)}$$

Motivation: bilevel programming

BKP is a bilevel integer programming problem:

$$\begin{aligned} \min \quad & c^T x + d^T y \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n \\ & y \in \arg\max \left\{ f^T y : Gx + Hy \leq g, y \in \mathbb{Z}^p \right\} \end{aligned} \tag{BIP}$$

y must be optimal for a second optimization problem (depending on x).

History and motivation: bilevel programming

Why bilevel?

- Competing parties (military, business)
- Semi-cooperating parties (federal and regional governments)
- A natural way to get harder versions of classical problems

History and motivation: bilevel knapsack

Complexity of BKP (Caprara, Carvalho, Lodi and Woeginger, 2014)

- Σ_2^p-complete
- no polysize IP formulation unless the polynomial hierarchy collapses
- no pseudopolynomial time algorithm unless P=NP

However...

History and motivation: bilevel knapsack

Complexity of BKP (Caprara, Carvalho, Lodi and Woeginger, 2014)

- Σ_2^p-complete
- no polysize IP formulation unless the polynomial hierarchy collapses
- no pseudopolynomial time algorithm unless P=NP

However...

- BKP is perhaps one of the "easiest" Σ_2^p-complete problems
- General bilevel solvers are far from the performance of problem-specific methods...

History and motivation: bilevel knapsack

Complexity of BKP (Caprara, Carvalho, Lodi and Woeginger, 2014)

- Σ_2^p-complete
- no polysize IP formulation unless the polynomial hierarchy collapses
- no pseudopolynomial time algorithm unless P=NP

However...

- BKP is perhaps one of the "easiest" Σ_2^p-complete problems
- General bilevel solvers are far from the performance of problem-specific methods... and this paper only widens that gap

History and motivation: bilevel knapsack

- DeNegre (2011) introduced the problem. Solved instances with ≤ 15 items
- Caprara, Carvalho, Lodi and Woeginger (2016): Solved instances with ≤ 50 items
- Tang, Richard and Smith (2016): Solved instances with ≤ 30 items
- Fischetti, Ljubic, Monaci, and Sinnl (2019). Solved instances with ≤ 55 items
- Lozano, Bergman and Cire (2022). Solved instances with ≤ 50 items
- Della Croce and Scatamacchia (2018). Solved instances with ≤ 500 items (henceforth the DCS algorithm)

...and more on approximation, complexity, problem variants, etc

Lower bounds and upper bounds

$$\begin{aligned} \min_{X\in\mathcal{U}} \quad & \sum_{i\in Y}^{n} p_i \\ \text{such that} \quad & Y \in \operatorname*{argmax}_{Y\in\mathcal{L}(X)} \sum_{i\in Y} p_i \end{aligned}$$

- Feasible solution $\implies$ upper bound
- Lower bounds are harder: first published 2018 (DCS algorithm)

Our contributions

All previous exact algorithms use MIP solvers.

We present a combinatorial algorithm which outperforms the previous best method, the DCS algorithm.

Our key insight: a new way of relaxing bilevel problems.

Preliminaries

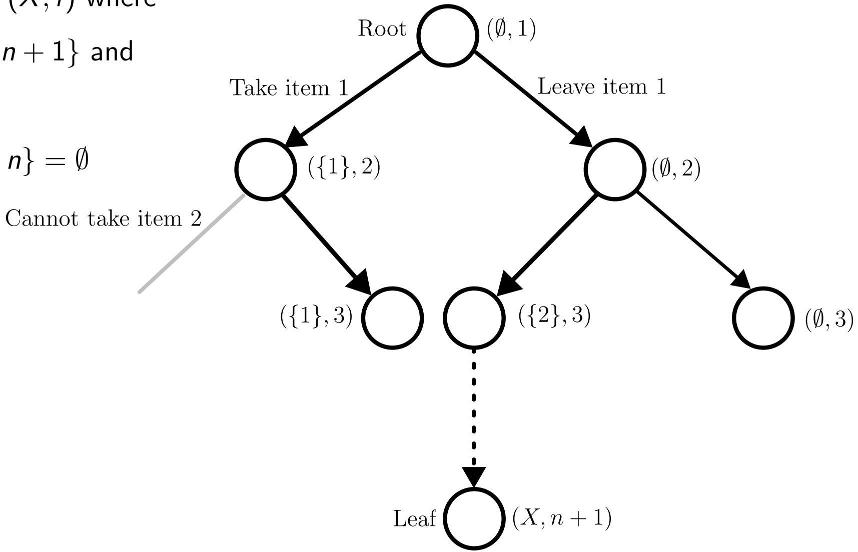
- X: upper level/leader's items
- Y: lower level/follower's items
- Items are enumerated by $\{1, \dots, n\}$ and ordered such that

$$\frac{p_1}{w_1^L} \geq \frac{p_2}{w_2^L} \geq \dots \geq \frac{p_n}{w_n^L}$$

Branch and bound

A node is a pair (X, i) where

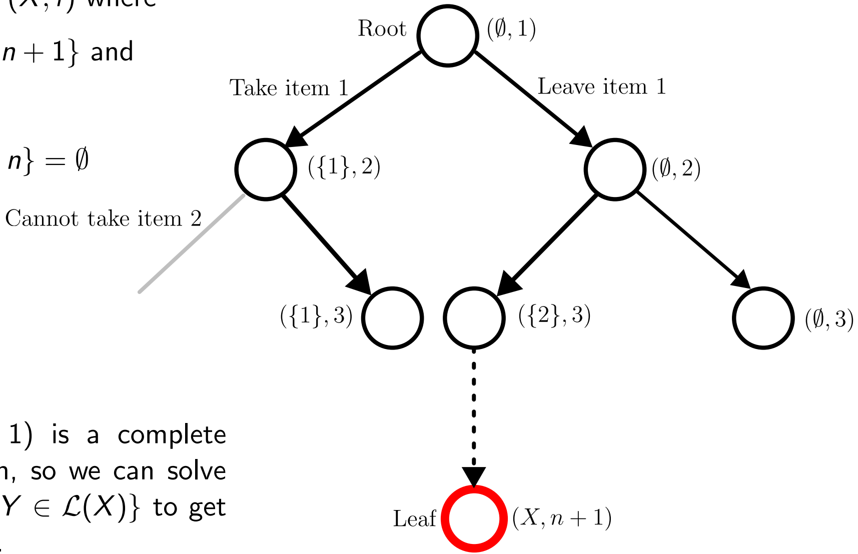
- $i \in \{1, \ldots, n+1\}$ and
- $X \in \mathcal{U}$
- $X \cap \{i, \ldots, n\} = \emptyset$



Branch and bound

A node is a pair (X, i) where

- $i \in \{1, \dots, n+1\}$ and
- $X \in \mathcal{U}$
- $X \cap \{i, \dots, n\} = \emptyset$

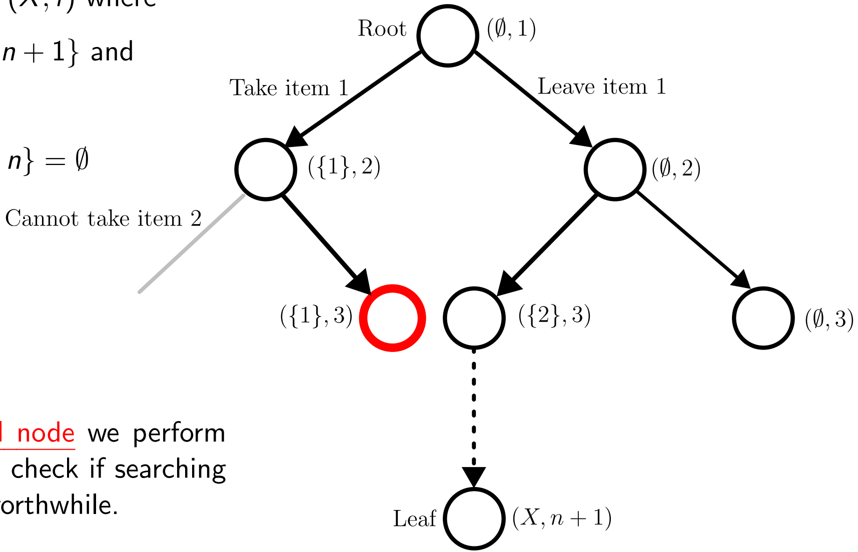


A leaf $(X, n+1)$ is a complete Leader's solution, so we can solve $\max\{\sum_{i \in Y} p_i : Y \in \mathcal{L}(X)\}$ to get an upper bound.

Branch and bound

A node is a pair (X, i) where

- $i \in \{1, \ldots, n+1\}$ and
- $X \in \mathcal{U}$
- $X \cap \{i, \ldots, n\} = \emptyset$



At each internal node we perform a bound test, to check if searching that branch is worthwhile.

Branch and bound: bound test

For node (X, i), is it possible that $\exists\, X' \supseteq X$ and $\exists\, Y' \in \mathcal{L}(X')$ with (X', Y') optimal?

If not, there is no point to explore the children of (X, i).

Branch and bound: bound test

For node (X, i), is it possible that $\exists\, X' \supseteq X$ and $\exists\, Y' \in \mathcal{L}(X')$ with (X', Y') optimal?

If not, there is no point to explore the children of (X, i).

To test this, we compute a lower bound at each node and test if it exceeds the current best upper bound.

Lower bound

Consider a node (X, i). In all descendants (X', i') of (X, i), we have $X' \supseteq X$ and $i' > i$. We want to lower bound the solution for such (X', i') which are leaves.

Lower bound

Consider a node (X, i). In all descendants (X', i') of (X, i), we have $X' \supseteq X$ and $i' > i$. We want to lower bound the solution for such (X', i') which are leaves.

We split this computation into two parts:

1. Lower bound using items $\{1, \ldots, i-1\}$ (prefix)
2. Lower bound using items $\{i, \ldots, n\}$ (postfix)

Lower bound

Consider a node (X, i). In all descendants (X', i') of (X, i), we have $X' \supseteq X$ and $i' > i$. We want to lower bound the solution for such (X', i') which are leaves.

We split this computation into two parts:

1. Lower bound using items $\{1, \dots, i-1\}$ (prefix)
2. Lower bound using items $\{i, \dots, n\}$ (postfix)

Note that:

- We know that the Leader uses capacity $\sum_{i \in X} w_i^U$ on the prefix
- This leaves $C^U - \sum_{i \in X} w_i^U$ for the postfix
- We don't know how much capacity the Follower uses on the prefix or the postfix, but any guess will give a lower bound.

Lower bound: items $\{1, \dots, i-1\}$ (prefix)

Given: Leader's items X, and (guessed) lower capacity c^L.

- X has already been decided on $\{1, \dots, i-1\}$
- So it suffices to find the optimal Y on items $\{1, \dots, i-1\} \setminus X$ with capacity c^L
- This is just a knapsack problem!

Lower bound: items $\{1, \ldots, i-1\}$ (prefix)

Given: Leader's items X, and (guessed) lower capacity c^L.

- X has already been decided on $\{1, \ldots, i-1\}$
- So it suffices to find the optimal Y on items $\{1, \ldots, i-1\} \setminus X$ with capacity c^L
- This is just a knapsack problem!

Definition

Let $K(S, c)$ denote the optimal objective value of the 0-1 knapsack problem with profits $(p_i)_{i \in S}$, weights $(w_i^L)_{i \in S}$ and capacity c.

- The desired bound is $K(\{1, \ldots, i-1\} \setminus X, c^L)$.

Lower bound: items $\{i, \ldots, n\}$ (postfix)

Given: remaining leader's capacity c^U, guessed follower's capacity c^L.

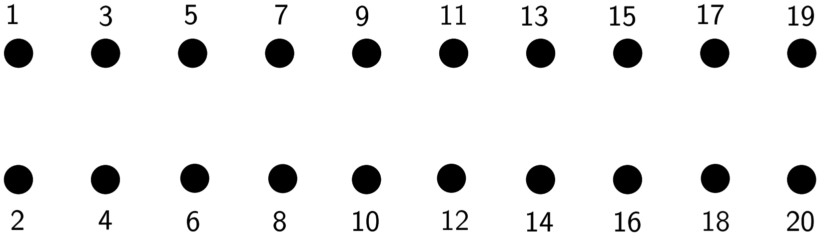
We relax the problem from bilevel to $2n$-level. The players alternative turns, considering one item at a time.

Lower bound: items $\{i, \ldots, n\}$ (postfix)

Given: remaining leader's capacity c^U, guessed follower's capacity c^L.

We relax the problem from bilevel to $2n$-level. The players alternative turns, considering one item at a time.

- **Round $2i-1$:** If $w_i^U + \sum_{j \in X} w_j^U \leq c^U$, Leader can add item i to X.



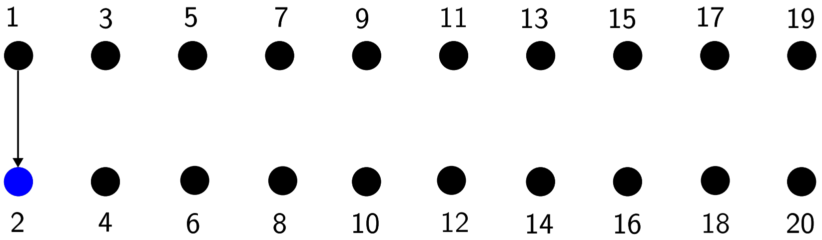
- **Round $2i$:** If $i \notin X$ and $w_i^L + \sum_{j \in Y} w_j^L \leq c^L$, Follower can add item i to Y.

Lower bound: items $\{i, \ldots, n\}$ (postfix)

Given: remaining leader's capacity c^U, guessed follower's capacity c^L.

We relax the problem from bilevel to $2n$-level. The players alternative turns, considering one item at a time.

- **Round 2i − 1:** If $w_i^U + \sum_{j \in X} w_j^U \leq c^U$, Leader can add item i to X.



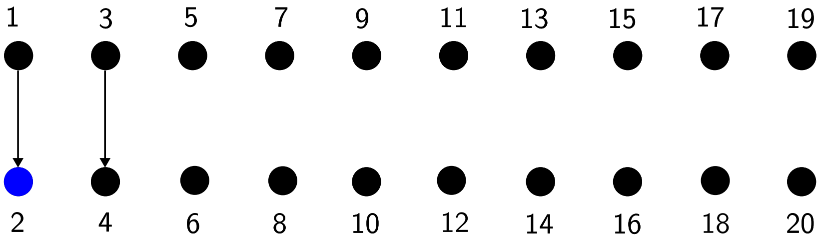
- **Round 2i:** If $i \notin X$ and $w_i^L + \sum_{j \in Y} w_j^L \leq c^L$, Follower can add item i to Y.

Lower bound: items $\{i, \ldots, n\}$ (postfix)

Given: remaining leader's capacity c^U, guessed follower's capacity c^L.

We relax the problem from bilevel to $2n$-level. The players alternative turns, considering one item at a time.

- **Round 2i − 1:** If $w_i^U + \sum_{j \in X} w_j^U \leq c^U$, Leader can add item i to X.



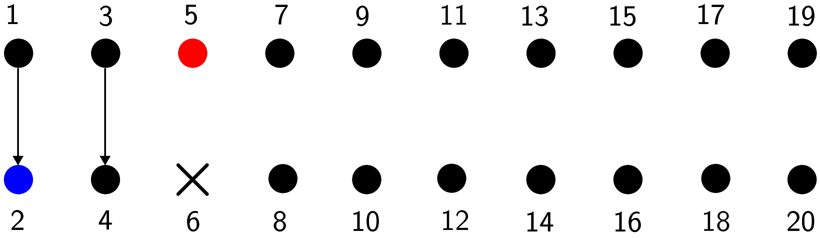
- **Round 2i:** If $i \notin X$ and $w_i^L + \sum_{j \in Y} w_j^L \leq c^L$, Follower can add item i to Y.

Lower bound: items $\{i, \dots, n\}$ (postfix)

Given: remaining leader's capacity c^U, guessed follower's capacity c^L.

We relax the problem from bilevel to $2n$-level. The players alternative turns, considering one item at a time.

- **Round 2i − 1:** If $w_i^U + \sum_{j \in X} w_j^U \leq c^U$, Leader can add item i to X.



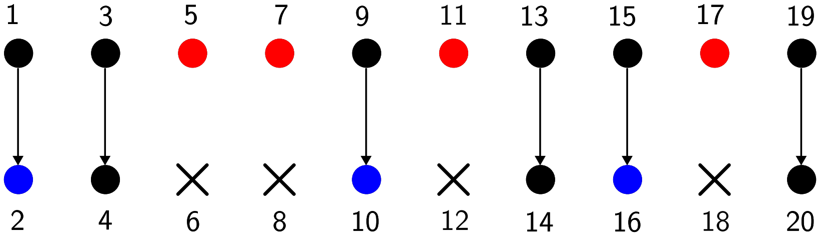
- **Round 2i:** If $i \notin X$ and $w_i^L + \sum_{j \in Y} w_j^L \leq c^L$, Follower can add item i to Y.

Lower bound: items $\{i, \ldots, n\}$ (postfix)

Given: remaining leader's capacity c^U, guessed follower's capacity c^L.

We relax the problem from bilevel to $2n$-level. The players alternative turns, considering one item at a time.

- **Round $2\mathbf{i}-1$:** If $w_i^U + \sum_{j \in X} w_j^U \leq c^U$, Leader can add item i to X.



- **Round 2i:** If $i \notin X$ and $w_i^L + \sum_{j \in Y} w_j^L \leq c^L$, Follower can add item i to Y.

What's changed? Intuition

- The Leader (minimizer) gets more information
- The Follower (maximizer) gets less information

Net result: a lower bound

Solving the modified game

The modified game admits a pseudopolytime algorithm by dynamic programming:

$$\omega(i, c^U, c^L) =$$
$$\begin{cases} \infty & \text{if } c^U < 0, \\ -\infty & \text{if } c^L < 0, \\ 0 & \text{if } c^U \geq 0, c^L \geq 0 \text{ and } i > n, \\ \min \left\{ \begin{matrix} \omega(i+1, c^U - w_i^U, c^L), \\ \max \left\{ \begin{matrix} \omega(i+1, c^U, c^L - w_i^L) + p_i, \\ \omega(i+1, c^U, c^L) \end{matrix} \right\} \end{matrix} \right\} & \text{if } c^U \geq 0,\ c^L \geq 0 \text{ and } i \leq n. \end{cases}$$

Theorem

$\omega(i, c^U, c^L)$ is the optimal objective value of the modified game when restricted to items $\{i, \ldots, n\}$ with Leader's capacity c^U and Follower's capacity c^L.

Postfix lower bound formalized

Definition

Let $OPT(i, c^U, c^L)$ be the optimal objective value for BKP when restricted to items $\{i, \ldots, n\}$ with Leader's capacity c^U and Follower's capacity c^L.

Theorem (Postfix lower bound)

For all $i \in [n]$, $0 \leq c^U \leq C^U$, and $0 \leq c^L \leq C^L$, we have

$$\omega(i, c^U, c^L) \leq OPT(i, c^U, c^L).$$

Combining prefix and postfix

Let $c \in [0, C^L]$ be a guess for how much capacity the Follower uses on the prefix. Recall:

- Prefix lower bound: $K(\{1, \ldots, i-1\} \setminus X, c)$
- Postfix lower bound: $\omega(i, C^U - \sum_{j \in X} w_j^U, C^L - c)$

Theorem

$$K(\{1, \ldots, i-1\} \setminus X, c) + \omega(i, C^U - \sum_{j \in X} w_j^U, C^L - c)$$

is a lower bound for node (X, i), and it can be computed in pseudopolynomial time.

Extensions & improvements: Solving trivial instances faster

Our lower bound is expensive: it requires pseudopolynomial time and memory.

Can we avoid computing it?

Extensions & improvements: Solving trivial instances faster

Our lower bound is expensive: it requires pseudopolynomial time and memory.

Can we avoid computing it?

Sometimes! Can get a much weaker lower bound in polytime by solving a linear program inspired by the DCS algorithm. Using this and a greedy upper bound, we can detect and solve trivial instances near instantly.

Extensions & improvements: sparse DP tables

Ok, but the lower bound is still expensive.

Can we compute less of it?

Extensions & improvements: sparse DP tables

Ok, but the lower bound is still expensive.

Can we compute less of it?

Yes! We can use sparse dynamic programming tables, like the classical DP-with-lists approach for knapsack.

This makes it practical to solve instances with arbitrarily large capacity.

Extensions & improvements: generalizations

Can this approach be applied to more problems?

Extensions & improvements: generalizations

Can this approach be applied to more problems?

<u>Yes!</u> Easy to generalize to:

- Bounded knapsack problem
- Multidimensional knapsack problem
- Min-max regret knapsack problem
- . . . hopefully many more

Implementation

- We implemented the algorithm in C++
- We reimplemented the DCS algorithm in C++ with Gurobi; our reimplementation generally matches or exceeds the performance of the original implementation
- DCS is parallelized via Gurobi
- In our algorithm, only the dynamic programming is parallelized
- We test on all instances from the literature, and generate some more

Computational results

Selected instances: generated by Fischetti, Monaci and Sinnl (2018), with n up to 500 and capacity up to 25000.

Class	DCS				Comb			
	#Opt	#Best	Avg	Max	#Opt	#Best	Avg	Max
uncorrelated	50	0	3.66	13.38	50	50	0.64	7.1
weak correlated	50	0	13.49	72.64	50	50	0.39	4.76
strong correlated*	41	0	689.58	3,600	50	50	0.46	5.02
inverse strong corr.*	38	0	919.91	3,600	50	50	1.17	31.11
almost strong corr.*	40	0	815.4	3,600	50	50	0.35	4.28
subset-sum*	35	0	1,087.18	3,600	42	42	588.57	3,600
even-odd subset-sum*	36	0	1,033.98	3,600	42	42	582.37	3,600
even-odd strong corr.*	41	0	747.12	3,600	50	50	0.73	17.06
similar weight uncorr.	50	0	22.89	79.85	50	50	0.12	0.35

(Running times in seconds)

Computational results

Selected instances: generated by Fischetti, Monaci and Sinnl (2018), with n up to 500 and capacity up to 25000.

Class	DCS #Opt	DCS #Best	DCS Avg	DCS Max	Comb #Opt	Comb #Best	Comb Avg	Comb Max
uncorrelated	50	0	3.66	13.38	50	50	0.64	7.1
weak correlated	50	0	13.49	72.64	50	50	0.39	4.76
strong correlated*	41	0	689.58	3,600	50	50	0.46	5.02
inverse strong corr.*	38	0	919.91	3,600	50	50	1.17	31.11
almost strong corr.*	40	0	815.4	3,600	50	50	0.35	4.28
subset-sum*	35	0	1,087.18	3,600	42	42	588.57	3,600
even-odd subset-sum*	36	0	1,033.98	3,600	42	42	582.37	3,600
even-odd strong corr.*	41	0	747.12	3,600	50	50	0.73	17.06
similar weight uncorr.	50	0	22.89	79.85	50	50	0.12	0.35

(Running times in seconds)

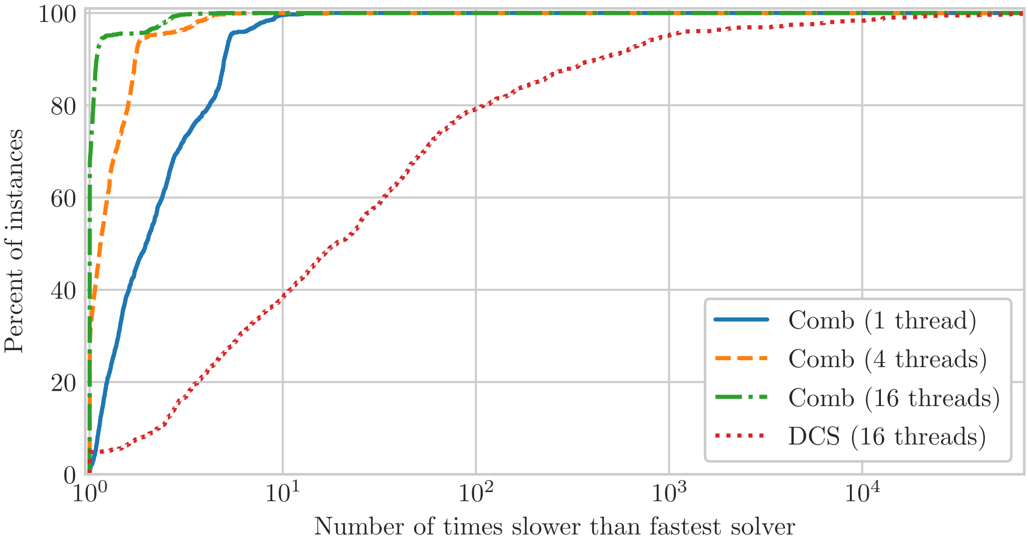
Computational results

Selected instances: generated by Fischetti, Monaci and Sinnl (2018), with n up to 500 and capacity up to 25000.

	DCS				Comb			
Class	#Opt	#Best	Avg	Max	#Opt	#Best	Avg	Max
uncorrelated	50	0	3.66	13.38	50	50	0.64	7.1
weak correlated	50	0	13.49	72.64	50	50	0.39	4.76
strong correlated*	41	0	689.58	3,600	50	50	0.46	5.02
inverse strong corr.*	38	0	919.91	3,600	50	50	1.17	31.11
almost strong corr.*	40	0	815.4	3,600	50	50	0.35	4.28
subset-sum*	35	0	1,087.18	3,600	42	42	588.57	3,600
even-odd subset-sum*	36	0	1,033.98	3,600	42	42	582.37	3,600
even-odd strong corr.*	41	0	747.12	3,600	50	50	0.73	17.06
similar weight uncorr.	50	0	22.89	79.85	50	50	0.12	0.35

(Running times in seconds)

Performance profile: all instances from the literature

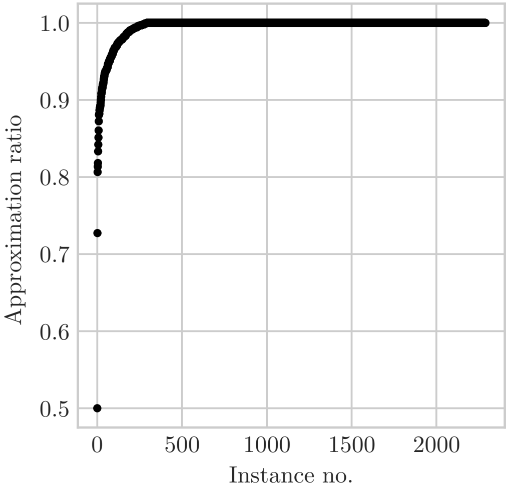


Lower bound strength in practice vs theory

- The relaxed game is optimal for BKP on 85% of instances
- There is a (contrived) family of instances where it has gap $O(n)$:

item no.	p	w^U	w^L
1	1	1	1
2	2	2	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-1$	$n-1$	$n-1$	$n-1$
n	$\binom{n}{2}$	$\binom{n}{2}$	$\binom{n}{2}+1$

- But with branch-and-bound, we solve this family near instantly



Conclusion

- Our solver has better performance on 99% of instances
- We solved 74% of the unsolved instances in the literature
- Key takeaway: relax the bilevel problem to $2n$ alternating levels: this gives a strong lower bound

Conclusion

- Our solver has better performance on 99% of instances
- We solved 74% of the unsolved instances in the literature
- Key takeaway: relax the bilevel problem to $2n$ alternating levels: this gives a strong lower bound

Future work / Open problems

- Is there a "fast" algorithm for subset-sum instances?
- What other problems would benefit from this type of relaxation?
- What can be said, theoretically, about the performance of our algorithm on particular instance classes?

Thanks for your attention!