Knapsack problem and variants

Michele Monaci

DEI, University of Bologna, Italy

16th ESICUP Meeting, ITAM, Mexico City, April 11, 2019

Overview

In memory of Egon Balas (1922-2019)

Professor Balas was a pioneer for Operations Research

Relevant contributions to disjunctive programming
but also to the area of Cutting & Packing.

In memory of Egon Balas (1922-2019)

Professor Balas was a pioneer for Operations Research
Relevant contributions to disjunctive programming
but also to the area of Cutting & Packing.

In memory of Egon Balas (1922-2019)

Professor Balas was a pioneer for Operations Research
Relevant contributions to disjunctive programming
but also to the area of Cutting & Packing.

In memory of Egon Balas (1922-2019)

FACETS OF THE KNAPSACK POLYTOPE*

Egon BALAS
Carnegie-Mellon University, Pittsburgh, Pa., U.S.A.

Received 23 October 1973
Revised manuscript received 20 May 1974

A necessary and sufficient condition is given for an inequality with coefficients 0 or 1 to define a facet of the knapsack polytope, i.e., of the convex hull of 0–1 points satisfying a given linear inequality. A sufficient condition is also established for a larger class of inequalities (with coefficients not restricted to 0 and 1) to define a facet for the same polytope, and a procedure is given for generating all facets in the above two classes. The procedure can be viewed as a way of generating cutting planes for 0–1 programs.

1. Introduction: Canonical equivalents of a linear inequality in 0–1 variables

Consider the inequality

$$\sum_{j\in N} a_j x_j \leqslant a_0 \,, \qquad (1)$$

where $a_0 > 0$, $a_j > 0$, and $x_j = 0$ or 1, $j \in N = \{1, \ldots, n\}$. Let N and all of its subsets to be considered below be ordered so that $a_j \geqslant a_{j+1}$, $j = 1, \ldots, n-1$.

A set $S \subset N$ will be called a *cover* or *covering set* for (1), if

(i) $$\sum_{j\in S} a_j > a_0 \,.$$

A cover for (1) will be called *minimal,* if

(ii) $$\sum_{j\in Q} a_j \leqslant a_0 \quad \text{for all proper subsets } Q \text{ of } S \,.$$

An Algorithm for Large Zero-One Knapsack Problems

EGON BALAS
Carnegie-Mellon University

EITAN ZEMEL
Northwestern University

(Received September 1977; accepted December 1979

We describe an algorithm for the 0-1 knapsack problem (KP), which relies mainly on three new ideas. The first one is to focus on what we call the core of the problem, namely, a knapsack problem equivalent to KP, defined on a particular subset of the variables. The size of this core is usually a small fraction of the full problem size, and does not seem to increase with the latter. While the core cannot be identified without solving KP, a satisfactory approximation can be found by solving the associated linear program (LKP). The second new ingredient is a binary search-type procedure for solving LKP which, unlike earlier methods, does not require any ordering of the variables. The computational effort involved in this procedure is linear in the number of variables. Finally, the third new feature is a simple heuristic which under certain conditions finds an optimal solution with a probability that increases with the size of KP. Computational experience with an algorithm based on the above ideas, on several hundred randomly generated test problems with 1,000–10,000 variables and with coefficients ranging from between 10 and 100 to between 10 and 10,000, indicates that for such problems the computational effort grows linearly with the number of variables and less than linearly with the range of coefficients. Average time per problem was less than a second, and the maximum time for any single problem was 3 seconds. Value-independent 0-1 knapsack problems (also randomly generated), were solved with a specialized version of the code in less than one-third of the time required for general 0-1 knapsack problems. To conclude, we identify a class of hard knapsack problems.

The Knapsack Problem

(1-dimensional) Knapsack Problem

Definition of the problem

Input data

- A set $N = \{1, \ldots, n\}$ of items, the j-th with a profit p_j and weight w_j;
- a knapsack with capacity C.

Problem

Select a set $S \subseteq N$ of items such that

- the total weight of the items in S does not exceed the capacity;
- the total profit of the selected items is a maximum.

(1-dimensional) Knapsack Problem

Definition of the problem

Input data

- A set $N = \{1, \ldots, n\}$ of items, the j-th with a profit p_j and weight w_j;
- a knapsack with capacity C.

Problem

Select a set $S \subseteq N$ of items such that

- the total weight of the items in S does not exceed the capacity;
- the total profit of the selected items is a maximum.

Natural binary linear formulation

Variables

$$x_j = \begin{cases} 1 & \text{if item } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \qquad (j \in N)$$

Model

$$(\mathrm{KP}) \qquad \max \sum_{j \in N} p_j x_j \tag{1}$$

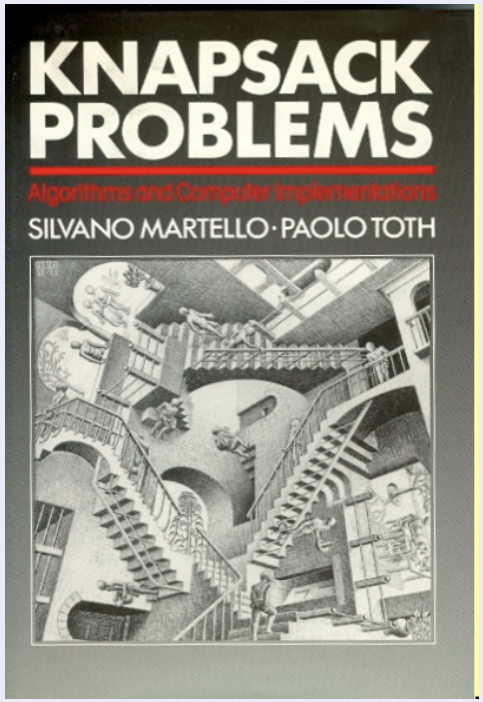
$$\sum_{j \in N} w_j x_j \leq C \tag{2}$$

$$x_j \in \{0, 1\} \qquad j \in N \tag{3}$$

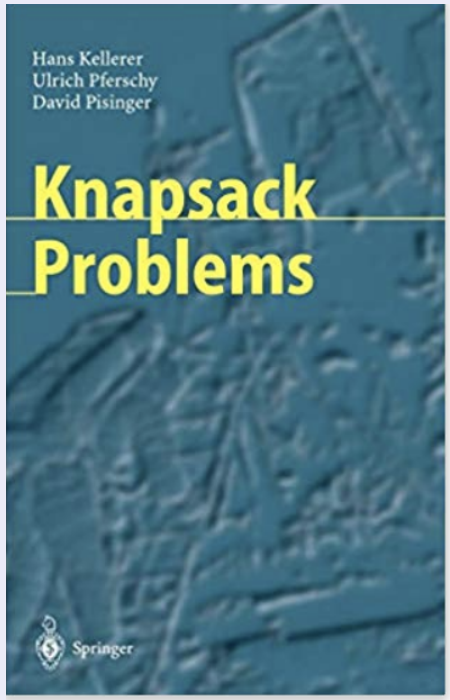
- efficiently solvable in practice;
- KP is (weakly) $\mathcal{NP}$-hard;
- alternative Linear Programming formulations with pseudo-polynomial size.

Literature

- 2 seminal books:
  Martello and Toth (1990)



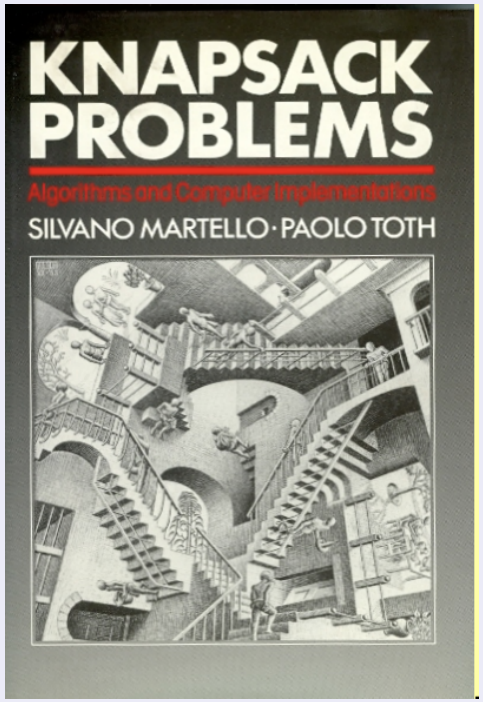
  Kellerer, Pferschy and Pisinger (2004)



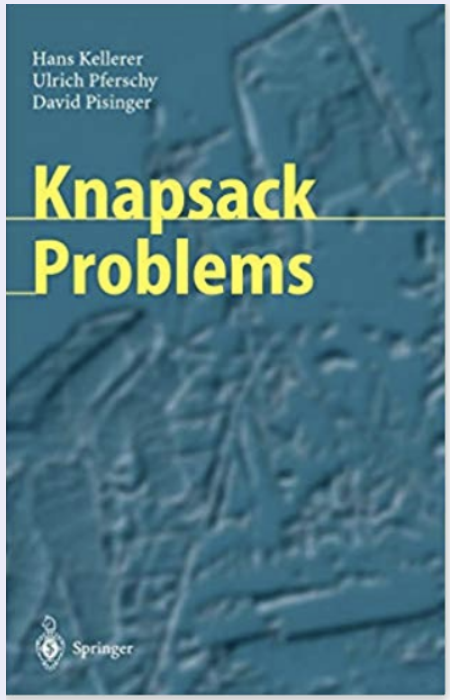
- Google scholar reports around 85,000 results when searching "knapsack problem";
- +16,000 results after 2015;

Literature

- 2 seminal books:
  Martello and Toth (1990)



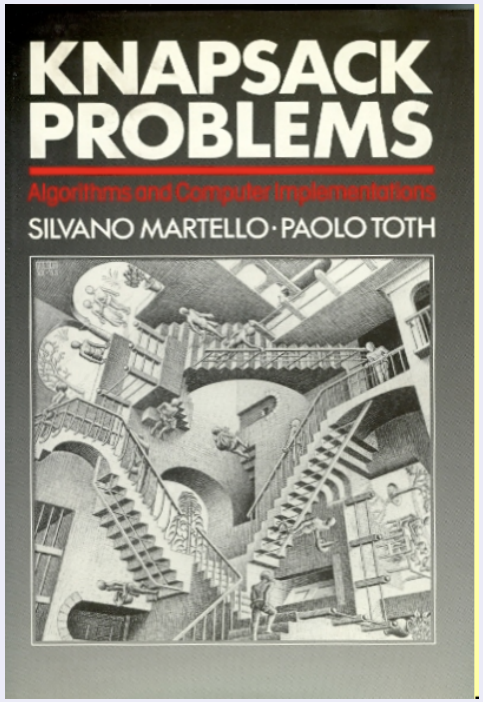
  Kellerer, Pferschy and Pisinger (2004)



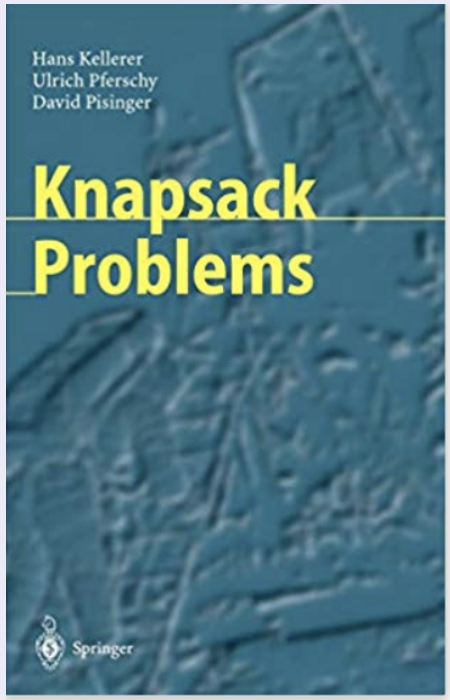
- Google scholar reports around 85,000 results when searching "knapsack problem";
- +16,000 results after 2015;

Literature

- 2 seminal books:
  Martello and Toth (1990)



  Kellerer, Pferschy and Pisinger (2004)



- Google scholar reports around 85,000 results when searching "knapsack problem";
- +16,000 results after 2015;

Why is KP so relevant?

- Easy problem to teach to students
- simplest combinatorial optimization problem
- resource allocation: portfolio selection, cut raw material, spectrum allocation in cognitive radio networks, ...
- cryptography: Merkle-Hellman key cryptosystem
- subproblem in many more complex combinatorial optimization problems

Good(?) news

We already have

- efficient algorithms for its exact solution;
- a clear picture of the LP relaxation;
- (Fully) Polynomial Time Approximation Schemes;
- ...

Does it still make sense to study knapsack problems?

Why is KP so relevant?

- Easy problem to teach to students
- simplest combinatorial optimization problem
- resource allocation: portfolio selection, cut raw material, spectrum allocation in cognitive radio networks, ...
- cryptography: Merkle-Hellman key cryptosystem
- subproblem in many more complex combinatorial optimization problems

Good(?) news

We already have

- efficient algorithms for its exact solution;
- a clear picture of the LP relaxation;
- (Fully) Polynomial Time Approximation Schemes;
- ...

Does it still make sense to study knapsack problems?

Why is KP so relevant?

- Easy problem to teach to students
- simplest combinatorial optimization problem
- resource allocation: portfolio selection, cut raw material, spectrum allocation in cognitive radio networks, ...
- cryptography: Merkle-Hellman key cryptosystem
- subproblem in many more complex combinatorial optimization problems

Good(?) news

We already have

- efficient algorithms for its exact solution;
- a clear picture of the LP relaxation;
- (Fully) Polynomial Time Approximation Schemes;
- ...

Does it still make sense to study knapsack problems?

Robust Knapsack problem

Robust Knapsack problem

Uncertainty of Input Values

For real-world problems input data is often not precise.
How to deal with this uncertainty?

1. stochastic optimization
2. robust optimization

Robust Optimization

(some of the) input parameters may attain arbitrary values
in a given interval.
Find a solution which remains feasible for any such input data.
cf. Soyster '73, Ben-Tal, Nemirovski '98

Robust Knapsack problem

Uncertainty of Input Values

For real-world problems input data is often not precise.
How to deal with this uncertainty?

1. stochastic optimization
2. robust optimization

Robust Optimization

(some of the) input parameters may attain arbitrary values
in a given interval.
Find a solution which remains feasible for any such input data.
cf. Soyster '73, Ben-Tal, Nemirovski '98

Robust Knapsack problem

Uncertainty of Input Values

For real-world problems input data is often not precise.
How to deal with this uncertainty?

1. stochastic optimization
2. robust optimization

Robust Optimization

(some of the) input parameters may attain arbitrary values
in a given interval.
Find a solution which remains feasible for any such input data.
cf. Soyster '73, Ben-Tal, Nemirovski '98

Robust Knapsack problem

Uncertainty of Input Values

For real-world problems input data is often not precise.
How to deal with this uncertainty?

1. stochastic optimization
2. robust optimization

Robust Optimization

(some of the) input parameters may attain arbitrary values
in a given interval.
Find a solution which remains feasible for any such input data.
cf. Soyster '73, Ben-Tal, Nemirovski '98

Robust Knapsack problem

Model of Bertsimas and Sim '04

- profits p_j are fixed
- weights belong to an interval $[w_j - \overline{w}_j, w_j + \overline{w}_j]$
- at most Γ weights can attain an arbitrary value in their interval, the remaining weights remain at w_j.

Γ denotes a "level of guarantee" for data uncertainty.

Assumptions:

- $C = 1$ normalization
- $w_j + \overline{w}_j \leq 1 \quad \forall\, j \in N$ trivial
- uncertainty is at most a constant fraction δ_j of the nominal weight, i.e., $\overline{w}_j = \delta_j\, w_j \quad \forall\, j \in N$

$$\Longrightarrow \quad w_j \leq \frac{1}{1+\delta_j} \quad \forall\, j \in N$$

Robust Knapsack problem

Model of Bertsimas and Sim '04

- profits p_j are fixed
- weights belong to an interval $[w_j - \overline{w}_j, w_j + \overline{w}_j]$
- at most Γ weights can attain an arbitrary value in their interval, the remaining weights remain at w_j.

Γ denotes a "level of guarantee" for data uncertainty.

Assumptions:

- $C = 1$ normalization
- $w_j + \overline{w}_j \leq 1 \quad \forall\, j \in N$ trivial
- uncertainty is at most a constant fraction δ_j of the nominal weight, i.e., $\overline{w}_j = \delta_j\, w_j \quad \forall\, j \in N$

$$\Longrightarrow \quad w_j \leq \frac{1}{1+\delta_j} \quad \forall\, j \in N$$

Robust Knapsack problem

Model of Bertsimas and Sim '04

- profits p_j are fixed
- weights belong to an interval $[w_j - \overline{w}_j, w_j + \overline{w}_j]$
- at most Γ weights can attain an arbitrary value in their interval, the remaining weights remain at w_j.

Γ denotes a "level of guarantee" for data uncertainty.

Assumptions:

- $C = 1$ normalization
- $w_j + \overline{w}_j \leq 1 \quad \forall\, j \in N$ trivial
- uncertainty is at most a constant fraction δ_j of the nominal weight, i.e., $\overline{w}_j = \delta_j\, w_j \quad \forall\, j \in N$

$$\Longrightarrow \quad w_j \leq \frac{1}{1+\delta_j} \quad \forall\, j \in N$$

Robust Knapsack problem

Model of Bertsimas and Sim '04

- profits p_j are fixed
- weights belong to an interval $[w_j - \overline{w}_j, w_j + \overline{w}_j]$
- at most Γ weights can attain an arbitrary value in their interval, the remaining weights remain at w_j.

Γ denotes a "level of guarantee" for data uncertainty.

Assumptions:

- $C = 1$ normalization
- $w_j + \overline{w}_j \leq 1 \quad \forall\, j \in N$ trivial
- uncertainty is at most a constant fraction δ_j of the nominal weight, i.e., $\overline{w}_j = \delta_j\, w_j \quad \forall\, j \in N$

$$\Longrightarrow \quad w_j \leq \frac{1}{1+\delta_j} \quad \forall\, j \in N$$

Robust Knapsack problem

Model of Bertsimas and Sim '04

- profits p_j are fixed
- weights belong to an interval $[w_j - \overline{w}_j, w_j + \overline{w}_j]$
- at most Γ weights can attain an arbitrary value in their interval, the remaining weights remain at w_j.

Γ denotes a "level of guarantee" for data uncertainty.

Assumptions:

- $C = 1$ normalization
- $w_j + \overline{w}_j \leq 1 \quad \forall\, j \in N$ trivial
- uncertainty is at most a constant fraction δ_j of the nominal weight, i.e., $\overline{w}_j = \delta_j\, w_j \quad \forall\, j \in N$

$\Longrightarrow \quad w_j \leq \frac{1}{1+\delta_j} \quad \forall\, j \in N$

Robust Knapsack problem

Model of Bertsimas and Sim '04

- profits p_j are fixed
- weights belong to an interval $[w_j - \overline{w}_j, w_j + \overline{w}_j]$
- at most Γ weights can attain an arbitrary value in their interval, the remaining weights remain at w_j.

Γ denotes a "level of guarantee" for data uncertainty.

Assumptions:

- $C = 1$ normalization
- $w_j + \overline{w}_j \leq 1 \quad \forall\, j \in N$ trivial
- uncertainty is at most a constant fraction δ_j of the nominal weight, i.e., $\overline{w}_j = \delta_j\, w_j \quad \forall\, j \in N$

$$\Longrightarrow \quad w_j \leq \frac{1}{1+\delta_j} \quad \forall\, j \in N$$

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Nominal (non-robust) solution:
pack items 1,2,3 $\Longrightarrow$ weight=12, profit = 21

Robust solution for $\Gamma = 1$ and $\delta_j = \frac{1}{2} \quad \forall j$:
items 1, 3 $\Longrightarrow$ robust weight=11, profit = 14
items 2, 3,4,5 $\Longrightarrow$ robust weight=12, profit = 16
items 3,. . . ,7 $\Longrightarrow$ robust weight=11, profit = 15

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Nominal (non-robust) solution:
pack items 1,2,3 $\Longrightarrow$ weight=12, profit = 21

Robust solution for $\Gamma = 1$ and $\delta_j = \frac{1}{2} \quad \forall j$:
items 1, 3 $\Longrightarrow$ robust weight=11, profit = 14
items 2, 3,4,5 $\Longrightarrow$ robust weight=12, profit = 16
items 3,. . . ,7 $\Longrightarrow$ robust weight=11, profit = 15

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Nominal (non-robust) solution:
pack items 1,2,3 $\Longrightarrow$ weight=12, profit = 21

Robust solution for $\Gamma = 1$ and $\delta_j = \frac{1}{2} \quad \forall j$:
items 1, 3 $\Longrightarrow$ robust weight=11, profit = 14
items 2, 3,4,5 $\Longrightarrow$ robust weight=12, profit = 16
items 3,. . . ,7 $\Longrightarrow$ robust weight=11, profit = 15

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Nominal (non-robust) solution:
pack items 1,2,3 $\Longrightarrow$ weight=12, profit = 21

Robust solution for $\Gamma = 1$ and $\delta_j = \frac{1}{2} \quad \forall j$:
items 1, 3 $\Longrightarrow$ robust weight=11, profit = 14
items 2, 3,4,5 $\Longrightarrow$ robust weight=12, profit = 16
items 3,. . . ,7 $\Longrightarrow$ robust weight=11, profit = 15

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Nominal (non-robust) solution:
pack items 1,2,3 $\Longrightarrow$ weight=12, profit = 21

Robust solution for $\Gamma = 1$ and $\delta_j = \frac{1}{2} \quad \forall j$:
items 1, 3 $\Longrightarrow$ robust weight=11, profit = 14
items 2, 3,4,5 $\Longrightarrow$ robust weight=12, profit = 16
items 3,. . . ,7 $\Longrightarrow$ robust weight=11, profit = 15

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Nominal (non-robust) solution:
pack items 1,2,3 $\Longrightarrow$ weight=12, profit = 21

Robust solution for $\Gamma = 1$ and $\delta_j = \frac{1}{2} \quad \forall j$:
items 1, 3 $\Longrightarrow$ robust weight=11, profit = 14
items 2, 3,4,5 $\Longrightarrow$ robust weight=12, profit = 16
items 3,. . . ,7 $\Longrightarrow$ robust weight=11, profit = 15

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example cnt.

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Robust solution for $\Gamma = 2$ and $\delta_j = 1 \quad \forall j$:
item 1 $\Longrightarrow$ robust weight=12, profit = 11
items 2,3 $\Longrightarrow$ robust weight=12, profit = 10
items 3,. . . ,6 $\Longrightarrow$ robust weight=12, profit = 12

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example cnt.

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Robust solution for $\Gamma = 2$ and $\delta_j = 1 \quad \forall j$:
item 1 $\Longrightarrow$ robust weight=12, profit = 11
items 2,3 $\Longrightarrow$ robust weight=12, profit = 10
items 3,. . . ,6 $\Longrightarrow$ robust weight=12, profit = 12

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example cnt.

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Robust solution for $\Gamma = 2$ and $\delta_j = 1 \quad \forall j$:
item 1 $\Longrightarrow$ robust weight=12, profit = 11
items 2,3 $\Longrightarrow$ robust weight=12, profit = 10
items 3,. . . ,6 $\Longrightarrow$ robust weight=12, profit = 12

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example cnt.

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Robust solution for $\Gamma = 2$ and $\delta_j = 1 \quad \forall j$:
item 1 $\Longrightarrow$ robust weight=12, profit = 11
items 2,3 $\Longrightarrow$ robust weight=12, profit = 10
items 3,. . . ,6 $\Longrightarrow$ robust weight=12, profit = 12

Robust Knapsack problem, Model of Bertsimas and Sim '04

Example cnt.

Given: 7 items, capacity $c = 12$

j	1	2	3, . . . ,7
p_j	11	7	3
w_j	6	4	2

Robust solution for $\Gamma = 2$ and $\delta_j = 1 \quad \forall j$:
item 1 $\Longrightarrow$ robust weight=12, profit = 11
items 2,3 $\Longrightarrow$ robust weight=12, profit = 10
items 3,. . . ,6 $\Longrightarrow$ robust weight=12, profit = 12

Robust Knapsack problem, Model of Bertsimas and Sim '04

Solution of the problem

- Mixed Integer Programming model (Bertsimas, Sim):
  The robust version of an optimization problem belongs to the same complexity class as the "nominal problem" with fixed input.
  MIP model for the robust knapsack problem based on duality.
- Iterate KP solution (Lee, Lee, Park, and Park):
  Guess the item k (say) with largest weight deviation that is included in the solution, and solve a KP with a modified capacity and items' weight.
  Iterate for $k = 1, 2, \ldots, n$ and return the best solution found.
- Dynamic programming (M., Pferschy & Serafini):
  a state $z(d, s, j)$ denotes the highest profit for a feasible solution with total weight d in which only items in $\{1, \ldots, s\}$ are considered and exactly s of them are taken with their upper weight bound $w_j + \overline{w}_j$.
  Time complexity $O(\Gamma n C)$, space complexity $O(n + \Gamma C)$.

Robust Knapsack problem, Model of Bertsimas and Sim '04

Solution of the problem

- Mixed Integer Programming model (Bertsimas, Sim):
  The robust version of an optimization problem belongs to the same complexity class as the "nominal problem" with fixed input.
  MIP model for the robust knapsack problem based on duality.
- Iterate KP solution (Lee, Lee, Park, and Park):
  Guess the item k (say) with largest weight deviation that is included in the solution, and solve a KP with a modified capacity and items' weight.
  Iterate for $k = 1, 2, \ldots, n$ and return the best solution found.
- Dynamic programming (M., Pferschy & Serafini):
  a state $z(d, s, j)$ denotes the highest profit for a feasible solution with total weight d in which only items in $\{1, \ldots, s\}$ are considered and exactly s of them are taken with their upper weight bound $w_j + \overline{w}_j$.
  Time complexity $O(\Gamma n C)$, space complexity $O(n + \Gamma C)$.

Robust Knapsack problem, Model of Bertsimas and Sim '04

Solution of the problem

- Mixed Integer Programming model (Bertsimas, Sim):
  The robust version of an optimization problem belongs to the same complexity class as the "nominal problem" with fixed input.
  MIP model for the robust knapsack problem based on duality.
- Iterate KP solution (Lee, Lee, Park, and Park):
  Guess the item k (say) with largest weight deviation that is included in the solution, and solve a KP with a modified capacity and items' weight.
  Iterate for $k = 1, 2, \ldots, n$ and return the best solution found.
- Dynamic programming (M., Pferschy & Serafini):
  a state $z(d, s, j)$ denotes the highest profit for a feasible solution with total weight d in which only items in $\{1, \ldots, s\}$ are considered and exactly s of them are taken with their upper weight bound $w_j + \overline{w}_j$.
  Time complexity $O(\Gamma nC)$, space complexity $O(n + \Gamma C)$.

Robust Knapsack problem, Model of Bertsimas and Sim '04

Computational experiments

- random (hard) KP instances with $n \leq 5000$;
- coefficients $\delta_j \in [1, \frac{1}{1+w_j}]$, different values of Γ;
- CPU seconds on an Intel i5-750 running at 2.67 GHz.

Γ	MILP	IT_KP	DP
1	9.325	0.081	0.007
10	2.336	0.021	0.016
50	0.271	0.008	0.050

Robust Knapsack problem, Model of Bertsimas and Sim '04

Computational experiments

- random (hard) KP instances with $n \leq 5000$;
- coefficients $\delta_j \in [1, \frac{1}{1+w_j}]$, different values of Γ;
- CPU seconds on an Intel i5-750 running at 2.67 GHz.

Γ	MILP	IT_KP	DP
1	9.325	0.081	0.007
10	2.336	0.021	0.016
50	0.271	0.008	0.050

Price of Robustness

Motivation

- Computational experiments show that the Robust Knapsack problem can be solved efficiently;
- What about the solution worsening caused by uncertainty?
- The solution worsening depends only on Γ and $\delta := \max_{j \in N} \delta_j$

For any instance I we denote:

$z(I)$ optimal solution value of nominal problem

$z^R_{\delta,\Gamma}(I)$ optimal solution value of robust problem

Worst-Case Ratio

price of robustness $R_{\delta,\Gamma} = \inf_I \dfrac{z^R_{\delta,\Gamma}(I)}{z(I)}$

Price of Robustness

Motivation

- Computational experiments show that the Robust Knapsack problem can be solved efficiently;
- What about the solution worsening caused by uncertainty?
- The solution worsening depends only on Γ and $\delta := \max_{j \in N} \delta_j$

For any instance I we denote:

$z(I)$ optimal solution value of nominal problem

$z^R_{\delta,\Gamma}(I)$ optimal solution value of robust problem

Worst-Case Ratio

price of robustness $R_{\delta,\Gamma} = \inf_I \frac{z^R_{\delta,\Gamma}(I)}{z(I)}$

Price of Robustness

Motivation

- Computational experiments show that the Robust Knapsack problem can be solved efficiently;
- What about the solution worsening caused by uncertainty?
- The solution worsening depends only on Γ and $\delta := \max_{j \in N} \delta_j$

For any instance I we denote:

$z(I)$ optimal solution value of nominal problem

$z^R_{\delta,\Gamma}(I)$ optimal solution value of robust problem

Worst-Case Ratio

price of robustness $\quad R_{\delta,\Gamma} = \inf_I \dfrac{z^R_{\delta,\Gamma}(I)}{z(I)}$

Price of Robustness: The Integer Case

Worst-Case Ratio

Theorem: For $\Gamma = 1$ we have:

$$R_{\delta,1} = \frac{1}{1 + \lceil \delta \rceil} \qquad \textit{for all} \quad \delta > 0.$$

Proof, upper bound:
Consider an instance with $n = 1 + \lceil \delta \rceil$ identical items

$$p_j = 1, \quad w_j = \frac{1}{1 + \lceil \delta \rceil}$$

- nominal solution: pack all items $\quad \Longrightarrow z = 1 + \lceil \delta \rceil$
- robust solution:
  robust weight of two items: $\frac{1}{1+\lceil \delta \rceil} + \frac{1}{1+\lceil \delta \rceil}(1+\delta) = \frac{2+\delta}{1+\lceil \delta \rceil} > 1$
  $\rightarrow$ pack only one item $\quad \Longrightarrow z^R = 1$
  $\rightarrow \frac{z^R}{z} = \frac{1}{1+\lceil \delta \rceil}$

Price of Robustness: The Integer Case

Worst-Case Ratio

Theorem: For $\Gamma = 1$ we have:

$$R_{\delta,1} = \frac{1}{1 + \lceil \delta \rceil} \qquad \textit{for all} \quad \delta > 0.$$

Proof, lower bound:
Consider the item set M in an optimal solution of the nominal problem.
Partition M into ℓ "robust" groups, sorted by nondecreasing weight:

- initially, each group corresponds to a single item;
- iteratively merge the first two groups, if this produces a robust feasible group;
- stop when no merge is possible.

It can be proved that $\ell \leq \lceil \delta \rceil + 1$.
Hence taking the group with maximum profit gives a robust solution with value

$$z^R \geq \frac{1}{\lceil \delta \rceil + 1} z$$

Price of Robustness: The Integer Case

Worst-Case Ratio

Theorem: For $\Gamma = 1$ we have:

$$R_{\delta,1} = \frac{1}{1 + \lceil \delta \rceil} \qquad \textit{for all} \quad \delta > 0.$$

Proof, lower bound:
Consider the item set M in an optimal solution of the nominal problem.
Partition M into ℓ "robust" groups, sorted by nondecreasing weight:

- initially, each group corresponds to a single item;
- iteratively merge the first two groups, if this produces a robust feasible group;
- stop when no merge is possible.

It can be proved that $\ell \leq \lceil \delta \rceil + 1$.
Hence taking the group with maximum profit gives a robust solution with value

$$z^R \geq \frac{1}{\lceil \delta \rceil + 1} z$$

Price of Robustness: The Integer Case

Worst-Case Ratio

Theorem: For $\Gamma \geq 2$ we have:

$$R_{\delta,\Gamma} = \frac{1}{1 + \lceil 2\delta \rceil} \qquad \textit{for all} \quad \delta > 0.$$

Proof, upper bound:
Consider an instance with $n = 1 + \lceil 2\delta \rceil$ identical items
$p_j = 1$, $w_j = \frac{1}{1+\lceil 2\delta \rceil}$

- nominal solution: pack all items $\quad \Longrightarrow z = 1 + \lceil 2\delta \rceil$
- robust solution:
  robust weight of two items: $2(1+\delta)\,\frac{1}{1+\lceil 2\delta \rceil} > 1$
  $\rightarrow$ pack only one item $\quad \Longrightarrow z^R = 1$
  $\rightarrow \frac{z^R}{z} = \frac{1}{1+\lceil 2\delta \rceil}$

Price of Robustness: The LP Relaxation

Worst-Case Ratio

price of robustness of the LP relaxation $\quad \overline{R}_{\delta,\Gamma} = \inf_I \dfrac{\overline{z}^R_{\delta,\Gamma}(I)}{\overline{z}(I)}$

where

$\overline{z}(I)$	optimal LP-relaxation of nominal problem
$\overline{z}^R_{\delta,\Gamma}(I)$	optimal LP-relaxation of robust problem

Worst-Case Ratio

Theorem:

$$\overline{R}_{\delta,\Gamma} = \frac{1}{1+\delta} \quad \textit{for all } \delta > 0,\ \Gamma \geq 1.$$

Robust Knapsack problem: LP Relaxation

Computational complexity

Theorem: The LP Relaxation of the Robust Knapsack problem can be computed in $O(n \log n)$ for any Γ.

Optimality gap

Theorem:

$$\lim_{\delta \to \infty} \inf_{I, |I|=n} \left\{ \frac{z^R_{\delta,\Gamma}(I)}{\overline{z}^R_{\delta,\Gamma}(I)} \right\} = \frac{1}{n} \qquad \text{for } \Gamma \in \{1, 2\}.$$

$$\frac{1}{n} \leq \lim_{\delta \to \infty} \inf_{I, |I|=n} \left\{ \frac{z^R_{\delta,\Gamma}(I)}{\overline{z}^R_{\delta,\Gamma}(I)} \right\} \leq \frac{\Gamma}{2n} \qquad \text{for all } \Gamma \geq 3.$$

Robust Knapsack problem: LP Relaxation

Computational complexity

Theorem: The LP Relaxation of the Robust Knapsack problem can be computed in $O(n \log n)$ for any Γ.

Optimality gap

Theorem:

$$\lim_{\delta \to \infty} \inf_{I, |I|=n} \left\{ \frac{z^R_{\delta,\Gamma}(I)}{\overline{z}^R_{\delta,\Gamma}(I)} \right\} = \frac{1}{n} \qquad \text{for } \Gamma \in \{1, 2\}.$$

$$\frac{1}{n} \leq \lim_{\delta \to \infty} \inf_{I, |I|=n} \left\{ \frac{z^R_{\delta,\Gamma}(I)}{\overline{z}^R_{\delta,\Gamma}(I)} \right\} \leq \frac{\Gamma}{2n} \qquad \text{for all } \Gamma \geq 3.$$

Robust Knapsack problem: Greedy heuristic

Robust Greedy (RG)

consider items in decreasing order of efficiency

insert current item into the solution set,
if the solution remains robust feasible
otherwise discard the item

iterate

also consider item with highest profit as singleton solution

Same as standard greedy with an extended feasibility check
$\Longrightarrow O(n \log \Gamma)$ running time (if items are sorted by efficiency).

Robust Knapsack problem: Greedy heuristic

Robust Greedy (RG)

consider items in decreasing order of efficiency

> insert current item into the solution set,
> if the solution remains robust feasible
> otherwise discard the item

iterate

also consider item with highest profit as singleton solution

Same as standard greedy with an extended feasibility check
$\Longrightarrow$ $O(n \log \Gamma)$ running time (if items are sorted by efficiency).

Robust Knapsack problem: Greedy heuristic

Robust Greedy (RG)

consider items in decreasing order of efficiency

> insert current item into the solution set,
> if the solution remains robust feasible
> otherwise discard the item

iterate

also consider item with highest profit as singleton solution

Same as standard greedy with an extended feasibility check
$\Longrightarrow O(n \log \Gamma)$ running time (if items are sorted by efficiency).

Greedy Algorithm for RKP

Analyze the performance of this approximation algorithm:

Definition

Worst-Case Performance Ratio of Greedy:

$$WR_{\delta,\Gamma}^{RG} = \inf_{I} \left\{ \frac{z_{\delta,\Gamma}^{RG}(I)}{z_{\delta,\Gamma}^{R}(I)} \right\}$$

Worst-Case Performance Ratio, Special Case $\Gamma = 1$

Theorem:

$$WR_{\delta,1}^{RG} = \frac{1}{2} \qquad \textit{for all } \delta < 1.$$

$$WR_{\delta,1}^{RG} = \frac{1}{1+\delta} \qquad \textit{for all } \delta \geq 1.$$

Worst-Case Performance Ratio, General Case $\Gamma \geq 2$

Theorem:

$$\frac{1}{2+\delta} \leq WR_{\delta,\Gamma}^{RG} \leq \frac{1}{1+\delta} \qquad \textit{for all } \delta > 0, \Gamma \geq 2.$$

Greedy Algorithm for RKP

Analyze the performance of this approximation algorithm:

Definition

Worst-Case Performance Ratio of Greedy:

$$WR^{RG}_{\delta,\Gamma} = \inf_{I} \left\{ \frac{z^{RG}_{\delta,\Gamma}(I)}{z^{R}_{\delta,\Gamma}(I)} \right\}$$

Worst-Case Performance Ratio, Special Case $\Gamma = 1$

Theorem:

$$WR^{RG}_{\delta,1} = \frac{1}{2} \qquad \textit{for all } \delta < 1.$$

$$WR^{RG}_{\delta,1} = \frac{1}{1+\delta} \qquad \textit{for all } \delta \geq 1.$$

Worst-Case Performance Ratio, General Case $\Gamma \geq 2$

Theorem:

$$\frac{1}{2+\delta} \leq WR^{RG}_{\delta,\Gamma} \leq \frac{1}{1+\delta} \qquad \textit{for all } \delta > 0, \Gamma \geq 2.$$

Greedy Algorithm for RKP

Analyze the performance of this approximation algorithm:

Definition

Worst-Case Performance Ratio of Greedy:

$$WR_{\delta,\Gamma}^{RG} = \inf_{I} \left\{ \frac{z_{\delta,\Gamma}^{RG}(I)}{z_{\delta,\Gamma}^{R}(I)} \right\}$$

Worst-Case Performance Ratio, Special Case $\Gamma = 1$

Theorem:

$$WR_{\delta,1}^{RG} = \frac{1}{2} \qquad \textit{for all } \delta < 1.$$

$$WR_{\delta,1}^{RG} = \frac{1}{1+\delta} \qquad \textit{for all } \delta \geq 1.$$

Worst-Case Performance Ratio, General Case $\Gamma \geq 2$

Theorem:

$$\frac{1}{2+\delta} \leq WR_{\delta,\Gamma}^{RG} \leq \frac{1}{1+\delta} \qquad \textit{for all } \delta > 0, \Gamma \geq 2.$$

Greedy Algorithm for RKP

Analyze the performance of this approximation algorithm:

Definition

Worst-Case Performance Ratio of Greedy:

$$WR_{\delta,\Gamma}^{RG} = \inf_{I} \left\{ \frac{z_{\delta,\Gamma}^{RG}(I)}{z_{\delta,\Gamma}^{R}(I)} \right\}$$

Worst-Case Performance Ratio, Special Case $\Gamma = 1$

Theorem:

$$WR_{\delta,1}^{RG} = \frac{1}{2} \qquad \textit{for all } \delta < 1.$$

$$WR_{\delta,1}^{RG} = \frac{1}{1+\delta} \qquad \textit{for all } \delta \geq 1.$$

Worst-Case Performance Ratio, General Case $\Gamma \geq 2$

Theorem:

$$\frac{1}{2+\delta} \leq WR_{\delta,\Gamma}^{RG} \leq \frac{1}{1+\delta} \qquad \textit{for all } \delta > 0, \Gamma \geq 2.$$

Interdiction Knapsack problem

Bilevel Games

In many real-world scenarios, a decision maker is not deciding alone . . .
her decisions must take decisions of other parties into account.

The resulting decision process can be modelled as a *two-player Stackelberg game*, in which two non-cooperative players (*leader* and *follower*) take their decisions in a sequential way:

- in the first round, the leader takes an action,
- in the second round, the follower reacts to it.

Follower decisions are influenced by the leader who possesses a complete knowledge of the follower optimization setting.

Bilevel Optimization

- challenging problems both in theory and in practice
- $\mathcal{NP}$-hard problems even in case both the leader and the follower are linear programs

Bilevel Games

In many real-world scenarios, a decision maker is not deciding alone . . .
her decisions must take decisions of other parties into account.

The resulting decision process can be modelled as a *two-player Stackelberg game*, in which two non-cooperative players (*leader* and *follower*) take their decisions in a sequential way:

- in the first round, the leader takes an action,
- in the second round, the follower reacts to it.

Follower decisions are influenced by the leader who possesses a complete knowledge of the follower optimization setting.

Bilevel Optimization

- challenging problems both in theory and in practice
- $\mathcal{NP}$-hard problems even in case both the leader and the follower are linear programs

Bilevel Games

In many real-world scenarios, a decision maker is not deciding alone . . .
her decisions must take decisions of other parties into account.

The resulting decision process can be modelled as a *two-player Stackelberg game*, in which two non-cooperative players (*leader* and *follower*) take their decisions in a sequential way:

- in the first round, the leader takes an action,
- in the second round, the follower reacts to it.

Follower decisions are influenced by the leader who possesses a complete knowledge of the follower optimization setting.

Bilevel Optimization

- challenging problems both in theory and in practice
- $\mathcal{NP}$-hard problems even in case both the leader and the follower are linear programs

Interdiction Games (IGs)

- special case of bilevel optimization problems
- leader and follower have **opposite objective functions**
- two-person, zero-sum sequential game
- studied mostly for network-based problems in the follower
- leader **interdicts** items of follower
  - type of interdiction: linear or **discrete**, cost increase or **destruction**
  - interdiction **budget**

Interdiction Games (IGs)

- special case of bilevel optimization problems
- leader and follower have **opposite objective functions**
- two-person, zero-sum sequential game
- studied mostly for network-based problems in the follower
- leader **interdicts** items of follower
  - type of interdiction: linear or **discrete**, cost increase or **destruction**
  - interdiction **budget**

(a) Linear, cost increase

(b) Discrete, destruction

Figure: Early Applications of Interdiction, following [Livy, 218BC]

Interdiction Knapsack problem

Corporate Strategy Problem:

- Companies A and B. A dominates the market. B wants to enter the market
- Whenever A and B target the same region, campaign of B is not effective
- Items are, e.g., demographic/geographic regions
- Cost and benefit for each target region
- A wants to prevent B in maximizing its gain (subject to available budget)

Mathematical Model

$$\min_{x} \max_{y} \; p^T y \tag{4}$$

$$v^T x \leq C_\ell \tag{5}$$

$$w^T y \leq C_f \tag{6}$$

$$x_j + y_j \leq 1, \qquad \forall j \in N \tag{7}$$

$$x, y \in \{0,1\}^n \tag{8}$$

Bilevel Knapsack is Σ_p^2-complete (Caprara, Carvalho, Lodi, Woeginger)

Interdiction Knapsack problem: reformulation

Single-level reformulation

The problem can be reformulated in the $n+1$ dimensional space as

$$\min_{x,\theta} \theta \tag{9}$$

$$\theta \geq \Phi(x) \tag{10}$$

$$v^T x \leq C_\ell \tag{11}$$

$$x \in \{0,1\} \tag{12}$$

where

$$\Phi(x) = \max_y \left\{ p^T y : w^T y \leq C_f, y \leq 1 - x, y \in \{0,1\} \right\}$$

is the response of the follower to a decision $\hat{x}$ of the leader.

Unfortunately

- constraints (10) are non-linear;
- function $\Phi(x)$ is neither convex nor concave.

Interdiction Knapsack problem: reformulation

Single-level reformulation

The problem can be reformulated in the $n+1$ dimensional space as

$$\min_{x,\theta} \theta \tag{9}$$

$$\theta \geq \Phi(x) \tag{10}$$

$$v^T x \leq C_\ell \tag{11}$$

$$x \in \{0,1\} \tag{12}$$

where

$$\Phi(x) = \max_y \left\{ p^T y : w^T y \leq C_f, y \leq 1 - x, y \in \{0,1\} \right\}$$

is the response of the follower to a decision $\hat{x}$ of the leader.

Unfortunately

- constraints (10) are non-linear;
- function $\Phi(x)$ is neither convex nor concave.

Interdiction Knapsack problem: computational experiments

Solution approaches

- CP: cutting planes approach (De Negre)
- CCLW: Specialized approach based on dualization of the follower's objective and on computing upper bounds on the problem (Caprara et al.)

Computational experiments

50 uncorrelated instances proposed by Caprara et al.
average CPU times over 10 instances (Xeon running at 2.66 GHz)

n	CP	CCLW
35	437.05	4.62
40	816.63	17.22
45	1423.81	40.06
50	1598.37	163.53
55	3600.00	766.27

"...for $n = 100$, computational times of one hour CPU Time . . . seem currently out of reach"

Interdiction Knapsack problem: reformulation

Wood's reformulation:
Find (sufficiently large) M_j's and reformulate the follower

$$\Phi(x) = \max\{p^T y - \sum_{j \in N} M_j x_j y_j : y \in Y\}$$

where $Y = \{y \in \{0,1\}^n : w^T y \leq C_f\}$ (independent of x).

Let $\hat{Y}$ be extreme points of $\operatorname{conv} Y$.

Benders-Like Reformulation

$$\min_{x,\theta} \theta \tag{13}$$

$$\theta \geq p^T \hat{y} - \sum_{j \in N} M_j x_j \hat{y}_j \qquad \forall \hat{y} \in \hat{Y} \tag{14}$$

$$v^T x \leq C_\ell \tag{15}$$

$$x \in \{0,1\} \tag{16}$$

Interdiction Knapsack problem: reformulation

Wood's reformulation:
Find (sufficiently large) M_j's and reformulate the follower

$$\Phi(x) = \max\{p^T y - \sum_{j \in N} M_j x_j y_j : y \in Y\}$$

where $Y = \{y \in \{0,1\}^n : w^T y \leq C_f\}$ (independent of x).

Let $\hat{Y}$ be extreme points of $\operatorname{conv} Y$.

Benders-Like Reformulation

$$\min_{x,\theta} \theta \tag{13}$$

$$\theta \geq p^T \hat{y} - \sum_{j \in N} M_j x_j \hat{y}_j \qquad \forall \hat{y} \in \hat{Y} \tag{14}$$

$$v^T x \leq C_\ell \tag{15}$$

$$x \in \{0,1\} \tag{16}$$

Benders-Like Reformulation

The Choice of M_j's is crucial

Large M_j's typically produce weak LP-relaxations

Theorem

For Interdiction Knapsack problem we can use $M_j = p_j$.

Approach

- Interdiction cuts $\theta \geq \sum_{j \in N} p_j(1 - x_j)\hat{y}_j \qquad \forall \hat{y} \in \hat{Y}$
- exponentially many constraints $\rightarrow$ Branch-and-Cut approach

Benders-Like Reformulation

The Choice of M_j's is crucial

Large M_j's typically produce weak LP-relaxations

Theorem

For Interdiction Knapsack problem we can use $M_j = p_j$.

Approach

- Interdiction cuts $\theta \geq \sum_{j \in N} p_j(1 - x_j)\hat{y}_j \qquad \forall \hat{y} \in \hat{Y}$
- exponentially many constraints $\rightarrow$ Branch-and-Cut approach

Branch-and-Cut

Preprocessing

$$x_a \geq x_b \qquad a, b \in N : w_a \leq w_b, v_a \leq v_b, p_a \geq p_b$$

Valid inequalities

Take any $\hat{y} \in \hat{Y}$. Then, the following inequalities are valid and there is no dominance among them.

$$\theta \geq \sum_{j \in N} p_j \hat{y}_j (1 - x_j) + (p_b - p_a)(1 - x_b) \qquad a, b \in N : \hat{y}_a = 1, \hat{y}_b = 0, p_a < p_b$$

$$\theta \geq \sum_{j \in N} p_j \hat{y}_j (1 - x_j) + p_b (x_a - x_b) \qquad a, b \in N : \hat{y}_a = 1, \hat{y}_b = 0, w_a \geq w_b$$

Can be extended to subset of items

Interdiction Knapsack problem: computational experiments

Computational experiments

50 uncorrelated instances proposed by Caprara et al.
average CPU times over 10 instances (comparable hardware)

n	CP	CCLW	B&C
35	437.05	4.62	0.32
40	816.63	17.22	0.34
45	1423.81	40.06	0.91
50	1598.37	163.53	1.17
55	3600.00	766.27	10.56

General purpose approach: similar results on different interdiction problems

Iterative MIP solution

Della Croce and Scatamacchia: new specific approach for interdiction knapsack

- a tight lower bound is obtained by guessing the critical item in the follower and solving a binary linear programming model;
- the algorithm exploits some expected features of an optimal solution of the classical knapsack problem (e.g., size of the core problem).

n	CP	CCLW	B&C	SD
35	437.05	4.62	0.32	0.14
40	816.63	17.22	0.34	0.16
45	1423.81	40.06	0.91	0.19
50	1598.37	163.53	1.17	0.18
55	3600.00	766.27	10.56	0.27

Tailored approach: similar results on larger instances

Non-Linear Generalized Assignment problem

Non-Linear Knapsack

Variant of the knapsack problem in which

- (some of) the variables are continuous
- the profit and weight associated with each item are described by non-linear functions of the variables

NLP Model

x_j = fraction of item j that is packed into the knapsack

$$\max \sum_{j \in N} p_j(x_j)$$

$$\sum_{j \in N} w_j(x_j) \leq C$$

$$0 \leq x_j \leq 1 \qquad j \in N$$

$$x_j \text{ integer} \qquad j \in I \subset N$$

Non-Linear Knapsack

Variant of the knapsack problem in which

- (some of) the variables are continuous
- the profit and weight associated with each item are described by non-linear functions of the variables

NLP Model

x_j = fraction of item j that is packed into the knapsack

$$\max \sum_{j \in N} p_j(x_j)$$

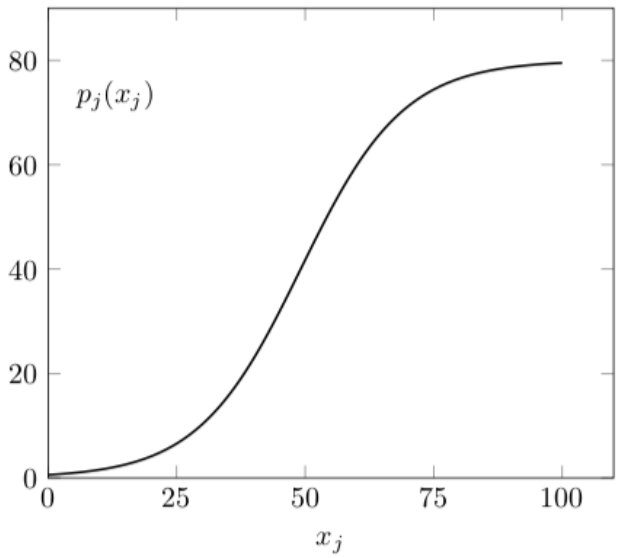
$$\sum_{j \in N} w_j(x_j) \leq C$$

$$0 \leq x_j \leq 1 \qquad j \in N$$

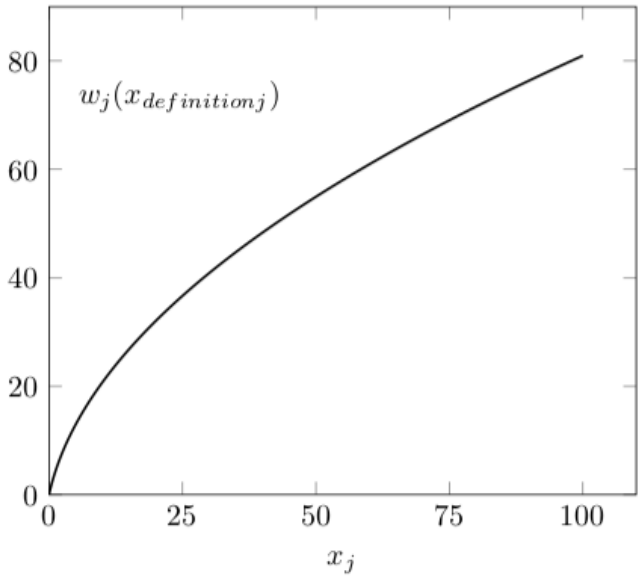
$$x_j \text{ integer} \qquad j \in I \subset N$$

Non-Linear Knapsack

profit function



weight function



Applications and literature

- financial models (Mathur, Salkin & Morito);
- production and inventory management (Ziegler);
- optimal design of queueing network models (Bitran & Tirupati);
- Different versions of the problem addressed in the literature (purely continuous, convex functions, quadratic functions, . . .)

Non-Linear Generalized Assignment problem

Definition of the problem

Input data

- A set $M = \{1, \ldots, m\}$ of knapsacks, the i-th with a capacity $c_i > 0$.
- A set $N = \{1, \ldots, n\}$ of items, the j-th with a maximum availability a_j.
- For each knapsack $i \in M$ and item $j \in N$ there is a profit function $p_{ij}(\cdot)$ and a weight function $w_{ij}(\cdot)$.

Problem

Pack (fraction of) items into the knapsacks so that

- the total weight of the items inserted in each knapsack does not exceed its capacity;
- the total amount of each item that is packed does not excedd its availability;
- the total profit of the packed items is a maximum;

Non-Linear Generalized Assignment problem

Definition of the problem

Input data

- A set $M = \{1, \ldots, m\}$ of knapsacks, the i-th with a capacity $c_i > 0$.
- A set $N = \{1, \ldots, n\}$ of items, the j-th with a maximum availability a_j.
- For each knapsack $i \in M$ and item $j \in N$ there is a profit function $p_{ij}(\cdot)$ and a weight function $w_{ij}(\cdot)$.

Problem

Pack (fraction of) items into the knapsacks so that

- the total weight of the items inserted in each knapsack does not exceed its capacity;
- the total amount of each item that is packed does not excedd its availability;
- the total profit of the packed items is a maximum;

NLGAP: NLP formulation

Variables

- x_{ij} fraction of item j that is packed into knapsack i;

Model

$$\text{(NLGAP)} \qquad \max \sum_{i \in M} \sum_{j \in N} p_{ij}(x_{ij}) \tag{17}$$

$$\sum_{j \in N} w_{ij}(x_{ij}) \leq c_i \qquad i \in M \tag{18}$$

$$\sum_{i \in M} x_{ij} \leq a_j \qquad j \in N \tag{19}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad i \in M; j \in N \tag{20}$$

where, for each $i \in M$ and $j \in N$:

$$u_{ij} := \max\{x : w_{ij}(x) \leq c_i\}$$

NLGAP: NLP formulation

Variables

- x_{ij} fraction of j that is packed into knapsack i;

Model

$$\max \sum_{i \in M} \sum_{j \in N} p_{ij}(x_{ij}) \tag{21}$$

$$\sum_{j \in N} w_{ij}(x_{ij}) \leq c_i \qquad i \in M \tag{22}$$

$$\sum_{i \in M} x_{ij} \leq a_j \qquad j \in N \tag{23}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad i \in M; j \in N \tag{24}$$

- The model is extremely hard in practice;
- NLP solvers typically fail in producing an optimal solution.

NLGAP: MILP approaches

Sampling

- Given an integer parameter s
- define $s+1$ distinct values (*samples*) for each pair (i,j)

$$0 = \alpha_{ij}^0 < \alpha_{ij}^1 < \ldots < \alpha_{ij}^s = u_{ij}.$$

Define and solve (Binary) Linear Programming models that

- depend on the samples, and
- produce either heuristic solutions or upper bounds.

NLGAP: MILP approaches

Sampling

- Given an integer parameter s
- define $s+1$ distinct values (*samples*) for each pair (i,j)

$$0 = \alpha_{ij}^0 < \alpha_{ij}^1 < \ldots < \alpha_{ij}^s = u_{ij}.$$

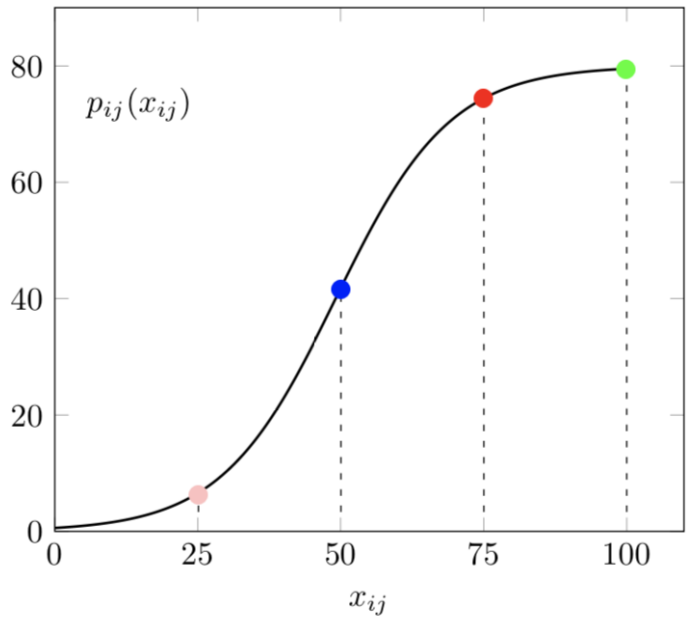
Define and solve (Binary) Linear Programming models that

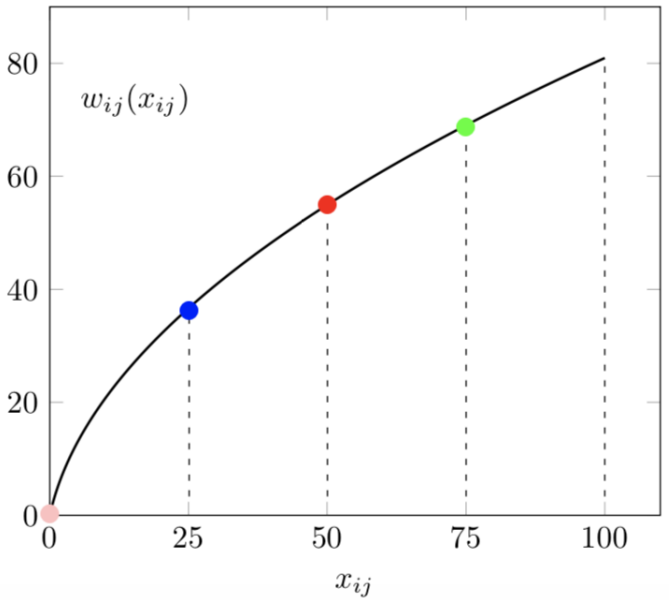
- depend on the samples, and
- produce either heuristic solutions or upper bounds.

NLGAP: Relaxation by Linear Programming

The relaxation is obtained by

- overestimating the values of the profit functions at each sample;
- underestimating the values of the weight functions at each sample.
- For each pair (k, j), variable x_{ij} is expressed as a convex combination of continuous variables ϑ_{ij}^k (one for each sample k).



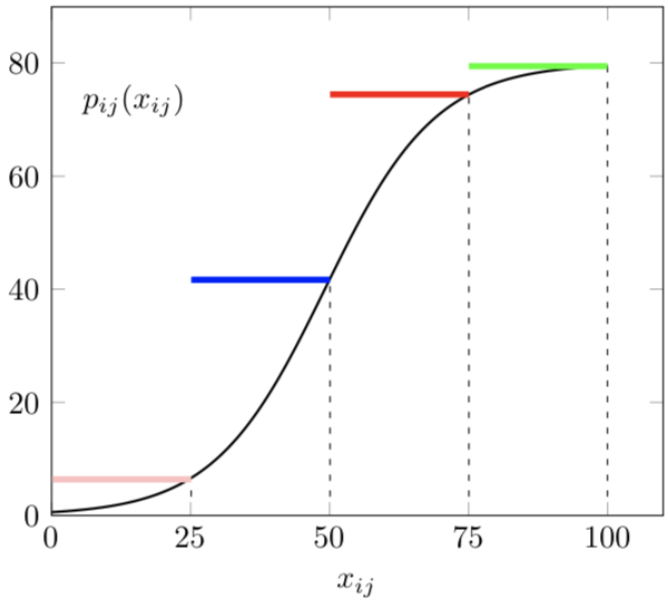


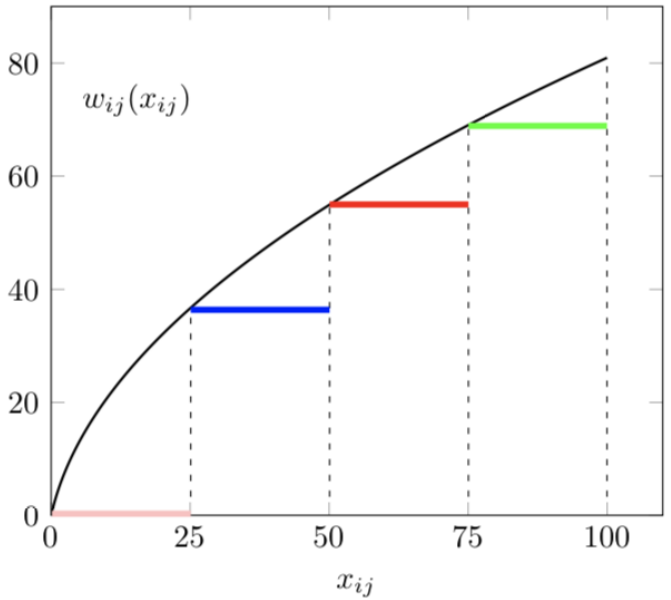
LP model $\rightarrow$ efficient solution even with a large number of samples

NLGAP: Relaxation by Binary Linear Programming

The relaxation is obtained by

- splitting the domain of each variable x_{ij} into intervals;
- each interval has the profit of its right extreme and the weight of its left extreme.



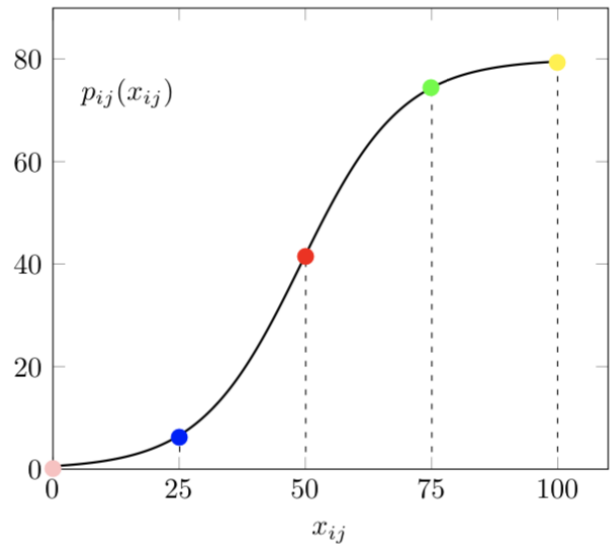


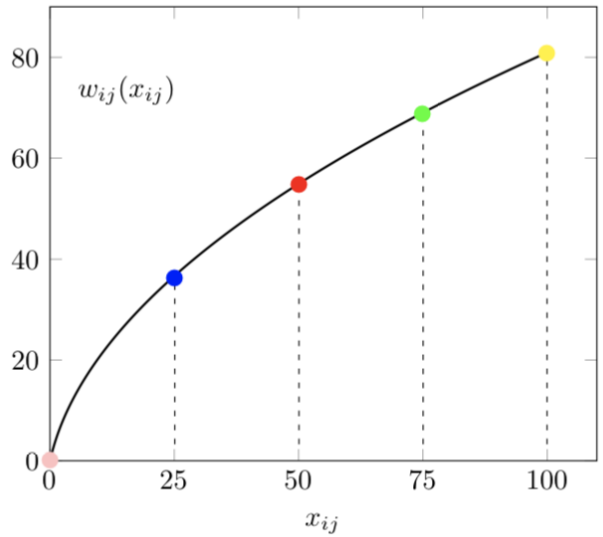
Computing the upper bound is an $\mathcal{NP}$-hard problem.

NLGAP: Heuristic by Binary Linear Programming

0-1 LP model in which

- each variable x_{ij} must attain a value that is exactly one of the samples;
- each sample has the correct (=sampled) profit and weight values.
- Classical (multiple) knapsack constraints.





Enhancements

- Integration with greedy heuristics;
- Local search post-optimization.

NLGAP: Computational experiments

Instances

- profit and weight functions taken from the literature:

$$\text{profit} \quad p_{ij}(x_{ij}) = \frac{c_{ij}}{1 + b_{ij}e^{-a_{ij}(x_{ij}+d_{ij})}} - \frac{c_{ij}}{1 + b_{ij}e^{-a_{ij}d_{ij}}}$$

$$\textit{Linear weight} \quad w_{ij}(x_{ij}) = w_{ij}x_{ij}$$

$$\textit{Nonlinear weight} \quad w_{ij}(x_{ij}) = \sqrt{p_{ij}x_{ij} + q_{ij}} - \sqrt{q_{ij}}$$

- Experiments on an Intel Xeon E5649 at 2.53 GHz using IBM ILOG CPLEX 12.6.0.0 as LP/ILP solver

Upper bounds

	LP		ILP	
# samples	Time	U/U^*	Time	U/U^*
5	0.22	1.836	87.35	1.292
10	0.39	1.294	106.13	1.104
50	2.22	1.069	173.75	1.022
100	12.66	1.051	196.69	1.014

NLGAP: Computational experiments

Lower bounds

Nonlinear solvers

- commercial solver Baron 18.11.12
- freeware solver Couenne 0.5

	Couenne (600)	Baron (600)	ILP (30)	Hybrid (30)
Nonlinear	48.667	34.465	2.673	2.380
Linear	60.420	41.007	2.855	2.324

- nonlinear solvers are dominated by the (KP-based) ILP heurstic
- hybrid algorithm computes (slightly) better solutions

Conclusions

Conclusions

Extensions

- Different definitions of robustness (e.g., Ben-Tal & Nemirowski)
- Two-dimensional Robust Knapsack problem
- Two-dimensional Interdiction Knapsack problem
- Two-dimensional Knapsack with non-linear constraints

Conclusions

- Many interesting and relevant variants and extensions of the Knapsack Problem
- Open problems both from a theoretical point of view and from a computational perspective

Does it still make sense to study knapsack problems?

Definitely YES

Conclusions

Extensions

- Different definitions of robustness (e.g., Ben-Tal & Nemirowski)
- Two-dimensional Robust Knapsack problem
- Two-dimensional Interdiction Knapsack problem
- Two-dimensional Knapsack with non-linear constraints

Conclusions

- Many interesting and relevant variants and extensions of the Knapsack Problem
- Open problems both from a theoretical point of view and from a computational perspective

Does it still make sense to study knapsack problems?

Definitely YES

Literature (1/2)

- A. Ben-Tal, A. Nemirovski, "Robust convex optimization", Mathematics of Operations Research (1998).
- D. Bertsimas, M. Sim, "The price of robustness", Operations Research (2004).
- G.R. Bitran, D. Tirupati, "Tradeoff curves, targeting and balancing in manufacturing queueing networks", Operations Research (1989).
- A. Caprara, M. Carvalho, A. Lodi, G.J. Woeginger, "A study on the computational complexity of the bilevel knapsack problem", SIAM Journal on Optimization (2014).
- A. Caprara, M. Carvalho, A. Lodi, G.J. Woeginger, "Bilevel knapsack with interdiction constraints", INFORMS Journal on Computing (2016).
- C. D'Ambrosio, S. Martello, M. Monaci, "Lower and upper bounds for the non-linear generalized assignment problem", Technical Report (2019).
- F. Della Croce, R. Scatamacchia, "A new exact approach for the Bilevel Knapsack with Interdiction Constraints", Technical Report (2019).
- S. De Negre, "Interdiction and discrete bilevel linear programming", PhD Thesis, Lehigh University (2011).
- M. Fischetti, I. Ljubić, M. Monaci, M. Sinnl, "Interdiction Games and Monotonicity with Application to Knapsack Problems", INFORMS Journal on Computing (2019).

Literature (2/2)

- C. Lee, K. Lee, K. Park, S. Park, "Technical Note – Branch-and-Price-and-Cut Approach to the Robust Network Design Problem Without Flow Bifurcations", Operations Research (2012).
- H. Kellerer, U. Pferschy, D. Pisinger, "Knapsack Problems", Springer (2004).
- S. Martello, P. Toth, "Knapsack Problems", John Wiley & Sons (1990).
- K. Mathur, H.M. Salkin, S. Morito, "A branch and search algorithm for a class of nonlinear knapsack problems", Operations Research Letters (1983).
- M. Monaci, U. Pferschy, "On the Robust Knapsack Problem", SIAM Journal on Optimization (2013).
- M. Monaci, U. Pferschy, P. Serafini, "Exact solution of the robust knapsack problem", Computers & Operations Research (2013).
- A.L. Soyster, "Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming", Operations Research (1973).
- H. Ziegler, "Solving certain singly constrained convex optimization problems in production planning", Operations Research Letters (1982).