ILP Formulations for the Lazy Bureaucrat Problem

F. Furini, I. Ljubić, M. Sinnl

PSL, Université Paris-Dauphine, 75775 Paris Cedex 16, France,
CNRS, LAMSADE UMR 7243

Department of Statistics and Operations Research,
University of Vienna, Vienna, Austria

EURO 2015, 12-15 July, Glasgow

Outline

Lazy Bureaucrat Problem

The lazy bureaucrat problem (LBP) is a scheduling problem (common arrivals and deadlines) in which a set of jobs has to be scheduled in the most inefficient way!

Definition

The lazy bureaucrat needs to choose a subset of jobs to execute in a single day, in a such a way that:

- no other job fits in his/her working hours
- the total profit of selected jobs (e.g., their duration) is minimized

Lazy Bureaucrat Problem

The lazy bureaucrat problem (LBP) is a scheduling problem (common arrivals and deadlines) in which a set of jobs has to be scheduled in the most inefficient way!

Definition

The lazy bureaucrat needs to choose a subset of jobs to execute in a single day, in a such a way that:

- no other job fits in his/her working hours
- the total profit of selected jobs (e.g., their duration) is minimized

Applications of the Lazy Bureaucrat Problem

Practical point of view

- how to distribute the available budget so that: (i) the minimal amount of money is allocated to funding requests and (ii) while having a good excuse that no additional funds can be granted without violating the available budget?

Theoretical point of view

- new interesting insights as the optimization is driven in the opposite direction when compared to their standard (non-lazy) counterparts

Applications of the Lazy Bureaucrat Problem

Practical point of view

- how to distribute the available budget so that: (i) the minimal amount of money is allocated to funding requests and (ii) while having a good excuse that no additional funds can be granted without violating the available budget?

Theoretical point of view

- new interesting insights as the optimization is driven in the opposite direction when compared to their standard (non-lazy) counterparts

Lazy Bureaucrat Problem – Mathematical definition

Definition (LBP with common arrivals and deadlines)

- We are given:
  - a set of jobs $I = \{1, \dots, n\}$
  - each job $i \in I$ has a duration $w_i \in \mathbb{N}$ and a profit $p_i \in \mathbb{N}$
  - all jobs arrive at the same time
  - all jobs have a common deadline $C \in \mathbb{N}$
- The goal is to find a least profitable subset of jobs $S^* \subset I$ such that:

$$S^* = \arg\min_{S \subset I} \{ \sum_{i \in S} p_i \mid \sum_{i \in S} w_i \leq C \text{ and } \sum_{i \in S} w_i + w_j > C, \; \forall j \notin S \}$$

- `time-spent` objective ($p_i = w_i$).
- `min-number-of-jobs` objective ($p_i = 1$).
- general objective function `weighted-sum`

Lazy Bureaucrat Problem – Mathematical definition

Definition (LBP with common arrivals and deadlines)

- We are given:
  - a set of jobs $I = \{1, \ldots, n\}$
  - each job $i \in I$ has a duration $w_i \in \mathbb{N}$ and a profit $p_i \in \mathbb{N}$
  - all jobs arrive at the same time
  - all jobs have a common deadline $C \in \mathbb{N}$
- The goal is to find a least profitable subset of jobs $S^* \subset I$ such that:

$$S^* = \arg\min_{S \subset I} \{\sum_{i \in S} p_i \mid \sum_{i \in S} w_i \leq C \text{ and } \sum_{i \in S} w_i + w_j > C, \ \forall j \notin S\}$$

- `time-spent` objective ($p_i = w_i$).
- `min-number-of-jobs` objective ($p_i = 1$).
- general objective function `weighted-sum`

Lazy Bureaucrat Problem – Mathematical definition

Definition (LBP with common arrivals and deadlines)

- We are given:
  - a set of jobs $I = \{1, \dots, n\}$
  - each job $i \in I$ has a duration $w_i \in \mathbb{N}$ and a profit $p_i \in \mathbb{N}$
  - all jobs arrive at the same time
  - all jobs have a common deadline $C \in \mathbb{N}$
- The goal is to find a least profitable subset of jobs $S^* \subset I$ such that:

$$S^* = \arg\min_{S \subset I} \{ \sum_{i \in S} p_i \mid \sum_{i \in S} w_i \leq C \text{ and } \sum_{i \in S} w_i + w_j > C, \ \forall j \notin S \}$$

- `time-spent` objective ($p_i = w_i$).
- `min-number-of-jobs` objective ($p_i = 1$).
- general objective function `weighted-sum`

Complexity issues

- The problem has been introduced in [1] where it was shown that a more general problem variant with individual arrival times and deadlines is NP-hard
- For the problem variant with common arrival times and deadlines, [3] show that the problem is weakly NP-hard for the `min-number-of-jobs` objective by reduction from subset-sum.
- Thus, the problem studied in this paper (with the more general `weighted-sum` objective) is also at least weakly NP-hard.
- In [5] a FPTAS have been proposed for the `time-spent` objective with common arrival and deadlines.

Property

The `weighted-sum` lazy bureaucrat problem with common arrivals and deadlines is weakly NP-hard

Relationship with the Knapsack Problem

- the LBP can be seen as the problem of packing a set of items in a knapsack in a most inefficient but feasible way
- jobs of I are the items
- job durations w_i are the item weights
- job profits p_i are the item profits
- the deadline C is the budget or capacity.

Notation and preprocessing

- items sorted in non-decreasing lexicographic order:
- non-decreasing weights $w_1 \leq w_2 \leq \cdots \leq w_n$
- non-decreasing profits p_i

- $C_i := C - w_i$, $C_1 \geq C_2 \geq \ldots C_n$.
- $w_{\max} := \max_{i \in I} w_i (= w_n)$
- $w_{\min} := \min_{i \in I} w_i (= w_1)$
- $W := \sum_{i \in I} w_i$
- $P := \sum_{i \in I} p_i$

SOLUTION PROPERTIES

Solution Properties

Definition (Maximal Knapsack Packing)

- A packing $\mathcal{P}$ is a set of items, with the property $\sum_{i \in \mathcal{P}} w_i \leq C$, i.e., a subset of items, which does not exceed the capacity C.
- A packing $\mathcal{P}$ is called maximal, iff $\mathcal{P} \cup \{i\}$ is not a packing for any $i \notin \mathcal{P}$.

Property

Each feasible LBP solution corresponds to a maximal feasible packing of the knapsack with capacity C.

Property

The capacity used by an arbitrary feasible solution S is bounded from below by $C - w_{\max} + 1$.

Solution Properties

Definition (Maximal Knapsack Packing)

- A packing $\mathcal{P}$ is a set of items, with the property $\sum_{i \in \mathcal{P}} w_i \leq C$, i.e., a subset of items, which does not exceed the capacity C.
- A packing $\mathcal{P}$ is called maximal, iff $\mathcal{P} \cup \{i\}$ is not a packing for any $i \notin \mathcal{P}$.

Property

Each feasible LBP solution corresponds to a maximal feasible packing of the knapsack with capacity C.

Property

The capacity used by an arbitrary feasible solution S is bounded from below by $C - w_{\max} + 1$.

Solution Properties

Definition (Maximal Knapsack Packing)

- A packing $\mathcal{P}$ is a set of items, with the property $\sum_{i \in \mathcal{P}} w_i \leq C$, i.e., a subset of items, which does not exceed the capacity C.
- A packing $\mathcal{P}$ is called maximal, iff $\mathcal{P} \cup \{i\}$ is not a packing for any $i \notin \mathcal{P}$.

Property

Each feasible LBP solution corresponds to a maximal feasible packing of the knapsack with capacity C.

Property

The capacity used by an arbitrary feasible solution S is bounded from below by $C - w_{\max} + 1$.

Solution Properties

Definition (Maximal Knapsack Packing)

- A packing $\mathcal{P}$ is a set of items, with the property $\sum_{i \in \mathcal{P}} w_i \leq C$, i.e., a subset of items, which does not exceed the capacity C.
- A packing $\mathcal{P}$ is called maximal, iff $\mathcal{P} \cup \{i\}$ is not a packing for any $i \notin \mathcal{P}$.

Property

Each feasible LBP solution corresponds to a maximal feasible packing of the knapsack with capacity C.

Property

The capacity used by an arbitrary feasible solution S is bounded from below by $C - w_{\max} + 1$.

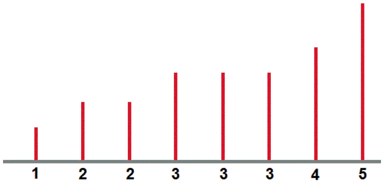
Critical Item (i)

We can only consider the minimum weight item outside the knapsack!

Definition (Critical Weight and Critical Item.)

$$i_c = \arg\min\{i \in I \mid \sum_{j \leq i} w_j > C\}$$

- the index of a critical item is the index of the first item that exceeds the capacity, assuming all $i \leq i_c$ will be taken as well.
- The critical weight (w_c) is the weight of the critical item ($w_c = w_{i_c}$).



Items after the critical cannot be minimum weight item outside!

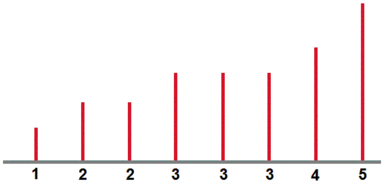
Critical Item (i)

We can only consider the minimum weight item outside the knapsack!

Definition (Critical Weight and Critical Item.)

$$i_c = \arg\min\{i \in I \mid \sum_{j \leq i} w_j > C\}$$

- the index of a critical item is the index of the first item that exceeds the capacity, assuming all $i \leq i_c$ will be taken as well.
- The critical weight (w_c) is the weight of the critical item ($w_c = w_{i_c}$).



Items after the critical cannot be minimum weight item outside!

Critical Item (ii)

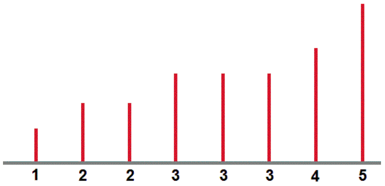
Proposition

The weight of the smallest item left out of any feasible LBP solution is bounded above by the critical weight w_c, i.e.:

$$S \text{ is feasible } \Rightarrow \min_{i \notin S} w_i \leq w_c.$$

Consequently, the size of the knapsack can be bounded from below as:

$$w(S) \geq C - w_c + 1.$$



Example

Weights(=profits) $\{1, 2, 2, 3, 3, 3, 4, 5\}$ and $C = 6$; $i_c = 4$ and $w_c = 3$

Critical Item (ii)

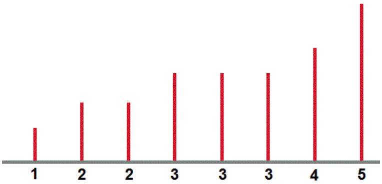
Proposition

The weight of the smallest item left out of any feasible LBP solution is bounded above by the critical weight w_c, i.e.:

$$S \text{ is feasible } \Rightarrow \min_{i \notin S} w_i \leq w_c.$$

Consequently, the size of the knapsack can be bounded from below as:

$$w(S) \geq C - w_c + 1.$$



Example

Weights(=profits) $\{1, 2, 2, 3, 3, 3, 4, 5\}$ and $C = 6$; $i_c = 4$ and $w_c = 3$

Main Observation

Observation

Once, the smallest item left out of the solution is known, the problem reduces to solving the knapsack problem with a lower and upper bound on its capacity (LU-KP)

If items are sorted in non-increasing order acc. to w_i and i is the smallest item outside:

$$(\mathrm{KP}_i) \quad J^* = \arg \min_{J \subseteq \{i+1,\dots,n\}} \{\sum_{j \in J} p_j + P_i \mid C - w_i - W_i + 1 \leq \sum_{j \in J} w_j \leq C - W_i\}$$

Based on it:

- FPTAS [5]
- Dynamic Programming ⇒ **We have a new $O(nC)$ approach!**
- ILP Models ⇒ Subject of this talk!
- CP Models
- Combinatorial lower bounds

Main Observation

Observation

Once, the smallest item left out of the solution is known, the problem reduces to solving the knapsack problem with a lower and upper bound on its capacity (LU-KP)

If items are sorted in non-increasing order acc. to w_i and i is the smallest item outside:

$$(\mathrm{KP}_i) \quad J^* = \arg \min_{J \subseteq \{i+1,\dots,n\}} \{\sum_{j \in J} p_j + P_i \mid C - w_i - W_i + 1 \leq \sum_{j \in J} w_j \leq C - W_i\}$$

Based on it:

- FPTAS [5]
- Dynamic Programming ⇒ **We have a new $O(nC)$ approach!**
- ILP Models ⇒ Subject of this talk!
- CP Models
- Combinatorial lower bounds

ILP FORMULATIONS

ILP Formulation 1

$$(\mathrm{ILP}_1) \qquad \min \sum_{i \in I} p_i x_i$$

$$\sum_{i \in I} w_i x_i \leq C \tag{0.1}$$

$$\sum_{j \in I, j \neq i} w_j x_j + w_i(1 - x_i) \geq (C + 1)(1 - x_i) \quad \forall i \in I : i \leq i_c \tag{0.2}$$

$$x_i \in \{0, 1\} \qquad \forall i \in I$$

Constraint (0.2) can be rewritten as

$$\sum_{j \in I, j \neq i} w_j x_j \geq (C_i + 1)(1 - x_i) \quad \forall i \in I : i \leq i_c.$$

where $C_i = C - w_i$.

ILP Formulation 1

$$(\text{ILP}_1) \qquad \min \sum_{i \in I} p_i x_i$$

$$\sum_{i \in I} w_i x_i \leq C \tag{0.1}$$

$$\sum_{j \in I, j \neq i} w_j x_j + w_i(1 - x_i) \geq (C + 1)(1 - x_i) \quad \forall i \in I : i \leq i_c \tag{0.2}$$

$$x_i \in \{0, 1\} \qquad \forall i \in I$$

Constraint (0.2) can be rewritten as

$$\sum_{j \in I, j \neq i} w_j x_j \geq (C_i + 1)(1 - x_i) \quad \forall i \in I : i \leq i_c.$$

where $C_i = C - w_i$.

Strengthening Covering Inequalities

Let (for a fixed $i \in I$)

$$\tilde{C} = \begin{cases} C_i + 1, & i \leq i_c \\ C_c + 1, & \text{otherwise} \end{cases}$$

Proposition

For a given $i \in I$, coefficients of the associated covering inequalities

$$\sum_{j \in I, j \neq i} w_j x_j + (C_i + 1) x_i \geq C_i + 1$$

can be down-lifted to $\sum_{k \in I} \alpha_k x_k \geq \tilde{C}$ where

$$\alpha_k := \begin{cases} \min\{w_c, \tilde{C}\}, & k = i \\ \min\{w_k, \tilde{C}\}, & \textit{otherwise} \end{cases} \qquad \forall k \in I$$

Global covering constraint:

$$\sum_{j \in I} w_i x_i \geq C_c + 1 \quad (C_c = C - w_c.) \tag{0.3}$$

(ILP_1): Computations

- (ILP_1) Basic
- (ILP_1^l) Lifted Variant
- In both cases: Branch-and-Cut approach - capacity constraints are separated on the fly

ILP Formulation 2

$$(\mathrm{ILP}_2) \qquad \min \sum_{i \in I} p_i x_i$$

$$\sum_{i \in I} w_i x_i \leq C \tag{0.4}$$

$$\sum_{i \in I} w_i x_i + z \geq C + 1 \tag{0.5}$$

$$z \leq w_c - (w_c - w_i)(1 - x_i) \qquad \forall i \in I, i \leq i_c \tag{0.6}$$

$$x \in \{0, 1\}, z \in \mathbb{N} \qquad \forall i \in I$$

- a single additional variable, but significantly simplified structure.

Formulations' structures

Cap	9		x_1	x_2	x_3	x_4	x_5	z
p_i			1	1	1	3	8	
w_i			1	1	1	3	8	
sol			1	1	1	1	0	4
	OPT	LP						
ILP_1	6	4.0588	0.6176	0.6176	0.6176	0.7353	0.0000	
ILP_1^l	6	4.1803	0.6885	0.6885	0.6885	0.7049	0.0000	
ILP_2	6	3.9718	0.7183	0.7183	0.7183	0.6056	0.0000	6.0282

Model I

```
Minimize
 obj:   x1 +   x2 +   x3 + 3 x4 + 8 x5
Subject To
 c1:    x1 +   x2 +   x3 + 3 x4 + 8 x5 <= 9
 c2: 9 x1 +   x2 +   x3 + 3 x4 + 8 x5 >= 9
 c3:    x1 + 9 x2 +   x3 + 3 x4 + 8 x5 >= 9
 c4:    x1 +   x2 + 9 x3 + 3 x4 + 8 x5 >= 9
 c5:    x1 +   x2 +   x3 + 7 x4 + 8 x5 >= 7
 c6:    x1 +   x2 +   x3 + 3 x4 + 2 x5 >= 2
Binaries
        x1     x2     x3     x4     x5
End
```

Model II

```
Minimize
 obj:     x1 +    x2 +    x3 + 3 x4 + 8 x5
Subject To
 c1:      x1 +    x2 +    x3 + 3 x4 + 8 x5     <= 9
 c2:      x1 +    x2 +    x3 + 3 x4 + 8 x5 + z >= 10
 c3: - 7 x1                                + z <= 1
 c4:           - 7 x2                      + z <= 1
 c5:                  - 7 x3               + z <= 1
 c6:                         - 5 x4        + z <= 3
 c7:                                         z <= 8
Binaries
          x1      x2      x3      x4     x5
End
```

IS THERE A CONNECTION

BETWEEN ILP_1 and ILP_2?

A (not so) surprising result!

Proposition

Benderization of ILP_2:
Projecting out z variables from ILP_2 results in the formulation ILP_1 in which constraints (0.2) are down-lifted as follows:

$$\sum_{j \in I, j \neq i} w_j x_j + w_c x_i \geq C + 1 - w_i$$

Corollary

If $\min\{w_k, C + 1 - w_i\} = w_k$, for all $k, i \in I$, then the LP-relaxations of ILP_1^l and ILP_2 are the same.

Observation

If $\exists i, k$ s.t. $w_k + w_i > C + 1$ ILP_1^l can still be stronger than ILP_2: coefficients next to x_j, $j \neq i$ are not down-lifted!

A (not so) surprising result!

Proposition

Benderization of ILP_2:
Projecting out z variables from ILP_2 results in the formulation ILP_1 in which constraints (0.2) are down-lifted as follows:

$$\sum_{j \in I, j \neq i} w_j x_j + w_c x_i \geq C + 1 - w_i$$

Corollary

If $\min\{w_k, C+1-w_i\} = w_k$, for all $k, i \in I$, then the LP-relaxations of ILP_1^l and ILP_2 are the same.

Observation

If $\exists i, k$ s.t. $w_k + w_i > C + 1$ ILP_1^l can still be stronger than ILP_2: coefficients next to x_j, $j \neq i$ are not down-lifted!

A (not so) surprising result!

Proposition

Benderization of ILP_2:
Projecting out z variables from ILP_2 results in the formulation ILP_1 in which constraints (0.2) are down-lifted as follows:

$$\sum_{j \in I, j \neq i} w_j x_j + w_c x_i \geq C + 1 - w_i$$

Corollary

If $\min\{w_k, C + 1 - w_i\} = w_k$, for all $k, i \in I$, then the LP-relaxations of ILP_1^l and ILP_2 are the same.

Observation

If $\exists i, k$ s.t. $w_k + w_i > C + 1$ ILP_1^l can still be stronger than ILP_2: coefficients next to x_j, $j \neq i$ are not down-lifted!

COMPUTATIONAL STUDY

Instances

- classical instance generator for 0/1 KP described in [6]
- two values: $\overline{R} \in \{1000, 10000\}$
- **small:** $n \in \{10, 20, 30, 40, 50, 100, 500, 1000, 2000\}$,
- **large:** $n \in \{7500, 10000, 15000, 20000\}$
- $C \in \{25\%, 50\%, 75\%\}$ of total weight
- 9 different groups, each with 54 instances $\Rightarrow$ 486 + 216 in total

Instances

1. Uncorrelated: w_j u.r. in $[1, \overline{R}]$, p_j u.r. in $[1, \overline{R}]$.
2. Weakly correlated: w_j u.r. in $[1, \overline{R}]$, p_j u.r. in $[w_j - \overline{R}/10, w_j + \overline{R}/10]$ so that $p_j \geq 1$.
3. Strongly correlated: w_j u.r. in $[1, \overline{R}]$, $p_j = w_j + \overline{R}/10$.
4. Inverse strongly correlated: p_j u.r. in $[1, \overline{R}]$, $w_j = p_j + \overline{R}/10$.
5. Almost strongly correlated: w_j u.r. in $[1, \overline{R}]$, p_j u.r. in $[w_j + \overline{R}/10 - \overline{R}/500, w_j + \overline{R}/10 + \overline{R}/500]$.
6. Subset-sum: w_j u.r. in $[1, \overline{R}]$, $p_j = w_j$.
7. Even-odd subset-sum: w_j even value u.r. in $[1, \overline{R}]$, $p_j = w_j$, c odd.
8. Even-odd strongly correlated: w_j even value u.r. in $[1, \overline{R}]$, $p_j = w_j + \overline{R}/10$, c odd.
9. Uncorrelated with similar weights: w_j u.r. in $[100\,\overline{R}, 100\,\overline{R} + \overline{R}/10]$, p_j u.r. in $[1, \overline{R}]$.

Greedy Heuristics

Table : Average percentage gaps from best known solutions by greedy heuristics.

	Class									
Algorithm	1	2	3	4	5	6	7	8	9	avg
Greedy heuristics										
greedy[$1/p_j$]	29.25	7.53	11.83	1.55	11.78	2.23	2.24	11.76	2.94	9.01
greedy[$1/w_j$]	66.85	9.08	11.83	1.55	11.78	2.23	2.24	11.76	55.35	19.19
greedy[w_j/p_j]	6.71	2.16	1.92	1.55	2.11	2.23	2.24	1.85	2.94	2.63
greedy[$1/(p_j * w_j)$]	56.20	8.35	11.83	1.55	11.78	2.23	2.24	11.76	2.94	12.10
greedy[$1/(p_j + w_j)$]	56.35	8.39	11.83	1.55	11.78	2.23	2.24	11.76	4.20	12.26
greedy[p_j/w_j]	71.34	19.67	11.83	22.03	11.78	2.23	2.24	11.76	68.07	24.55
greedy-comb	6.71	1.14	1.03	1.55	1.07	2.23	2.24	0.96	2.82	2.19

- Observe: sorting according to w_j/p_j performs the best, on average, which is also intuitive (we prefer items with low profit and high weight).

Formulation relative strength

Table : Average percentage gaps of the different linear programming relaxations.

	Class									
Algorithm	1	2	3	4	5	6	7	8	9	avg
Linear Relaxation										
ILP_1	20.11	41.52	40.09	45.62	39.90	45.35	45.35	40.17	22.98	37.90
ILP_1^l	13.67	3.19	2.75	2.52	2.68	2.92	2.93	2.84	6.64	4.46
ILP_2	13.67	3.19	2.79	2.52	2.68	2.95	2.96	2.88	6.64	4.47

- Lifting significantly reduces the LP-gap of (ILP_1)
- (ILP_2) provides very strong lower bounds, comparable to those obtained after lifting (ILP_1).
- The first class of instances (uncorrelated weights and profits) is by far the most difficult one, with average LP-gaps of more than 13% for all ILP formulations.

Formulation Comparison: Over all classes (**small**)

Items	ILP_1	ILP_1 ILP'_1	B&C	ILP_2	DP
Avg *t***[sec.s]**					
10	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0
30	0.2	0.1	0.1	0.1	0.0
40	0.4	0.2	0.2	0.2	0.1
50	0.9	0.2	0.7	0.3	0.1
100	34.7	2.9	19.5	1.4	0.3
500	32.1	8.8	70.1	0.5	1.3
1000	150.7	18.8	32.1	1.1	5.2
2000	105.1	73.8	107.0	7.5	8.9
AVG	19.9	10.5	14.2	1.2	1.7
# of TL					
10	0	0	0	0	0
20	0	0	0	0	0
30	0	0	0	0	0
40	0	0	0	0	0
50	0	0	0	0	0
100	2	0	1	0	0
500	24	4	27	5	3
1000	28	2	32	3	3
2000	44	9	36	5	5

Formulation Comparison

Table : Class 1-2-3

	Avg t[sec.s]			# of TL		
Items	ILP_1^l	ILP_2	DP	ILP_1^l	ILP_2	DP
7500		16.8	105.7	10	0	0
10000		38.8	191.2	10	0	0
15000		16.0	417.6	10	0	0
20000		58.4	737.5	10	0	0
7500		13.3	118.6	10	0	0
10000		28.6	206.7	10	0	0
15000		28.5	437.2	10	0	0
20000		2.3	700.1	10	0	0
7500	27.4	3.2	106.2	3	0	0
10000		7.7	171.1	10	0	0
15000		28.6	407.2	10	0	0
20000		5.3	658.5	10	0	0

Formulation Comparison

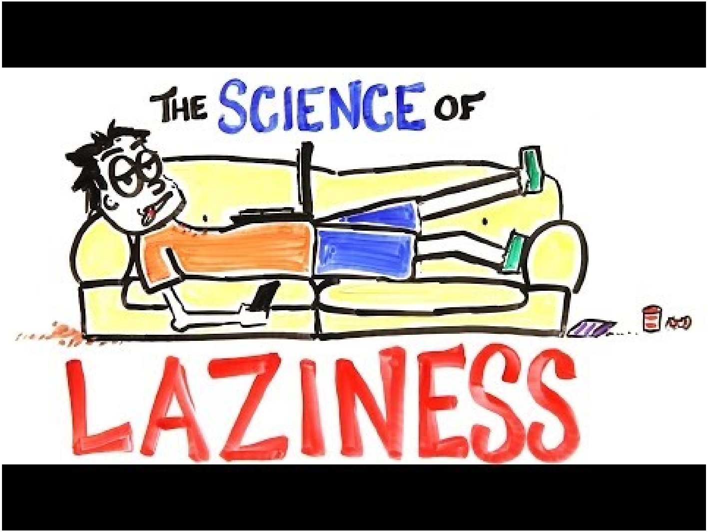
Table : Class 4-5-6

	Avg t[sec.s]			# of TL		
Items	ILP_1^l	ILP_2	DP	ILP_1^l	ILP_2	DP
7500	15.1	8.0	131.0	3	2	0
10000		2.9	238.1	10	2	0
15000		3.1	503.7	10	0	0
20000		15.1	643.3	10	1	1
7500		115.1	107.8	10	4	0
10000		127.1	192.8	10	2	0
15000		5.7	419.4	10	3	0
20000		3.8	701.6	10	2	0
7500		4.3	104.9	10	0	0
10000		4.4	183.4	10	1	0
15000		9.2	393.7	10	0	0
20000		59.6	687.3	10	0	0

Formulation Comparison

Table : Class 7-8-9

	Avg t[sec.s]			# of TL		
Items	ILP_1^l	ILP_2	DP	ILP_1^l	ILP_2	DP
7500		20.7	104.0	10	0	0
10000		5.5	185.1	10	0	0
15000		20.2	400.2	10	0	0
20000		15.6	679.2	10	0	0
7500	34.6	0.9	103.5	3	0	0
10000		2.6	198.6	10	0	0
15000		6.7	396.9	10	0	0
20000		2.3	683.9	10	0	0
7500	18.0	3.7	746.6	4	1	7
10000	32.6	0.3		2	3	10
15000		0.6		10	1	10
20000		0.7		10	6	10


THE SCIENCE OF
LAZINESS

Recap and future developments

- We studied ILP and DP approaches for solving the LBP problem to optimality. ILP_1^l can be obtained by "Benderization" of ILP_2
- Our computational study showed that the LBP is more difficult than the KP:
  - for the latter, instances with several thousands of items can be easily solved,
  - while for the LBP, a few thousands items make the problem difficult.
- **Greedy Boss:** Find a most profitable subset of jobs S^* to be executed so that the schedule exceeds the deadline, but after removing any of the scheduled jobs, the deadline is respected.
  - solution properties, valid inequalities, solution approaches
- **Lazy Bureaucrat vs Greedy Boss Games!**

Recap and future developments

- We studied ILP and DP approaches for solving the LBP problem to optimality. ILP_1^l can be obtained by "Benderization" of ILP_2
- Our computational study showed that the LBP is more difficult than the KP:
  - for the latter, instances with several thousands of items can be easily solved,
  - while for the LBP, a few thousands items make the problem difficult.
- **Greedy Boss:** Find a most profitable subset of jobs S^* to be executed so that the schedule exceeds the deadline, but after removing any of the scheduled jobs, the deadline is respected.
  - solution properties, valid inequalities, solution approaches
- **Lazy Bureaucrat vs Greedy Boss Games!**

References

E. M. Arkin, M. A. Bender, J. S. Mitchell, and S. S. Skiena.
The lazy bureaucrat scheduling problem.
Information and Computation, 184(1):129–146, 2003.

J. Boyar, L. Epstein, L. M. Favrholdt, J. S. Kohrt, K. S. Larsen, M. M. Pedersen, and S. Wøhlk.
The maximum resource bin packing problem.
Theoretical Computer Science, 362(1–3):127–139, 2006.

L. Gai and G. Zhang.
On lazy bureaucrat scheduling with common deadlines.
Journal of Combinatorial Optimization, 15(2):191–199, 2008.

D. Pisinger.
David Pisinger's optimization codes, 2014.
http://www.diku.dk/~pisinger/codes.html.

L. Gourvés, J. Monnot, and A. T. Pagourtzis.
The lazy bureaucrat problem with common arrivals and deadlines: Approximation and mechanism design.
In L. Gasieniec and F. Wolter, editors, Fundamentals of Computation Theory, volume 8070 of Lecture Notes in Computer Science, pages 171–182. Springer Berlin Heidelberg, 2013.

S. Martello, D. Pisinger, and P. Toth.
Dynamic programming and strong bounds for the 0-1 knapsack problem.
Management Science, 45:414–424, 1999.