University of Trieste
Architecture and Engineering Department

Master Degree in Computer Science

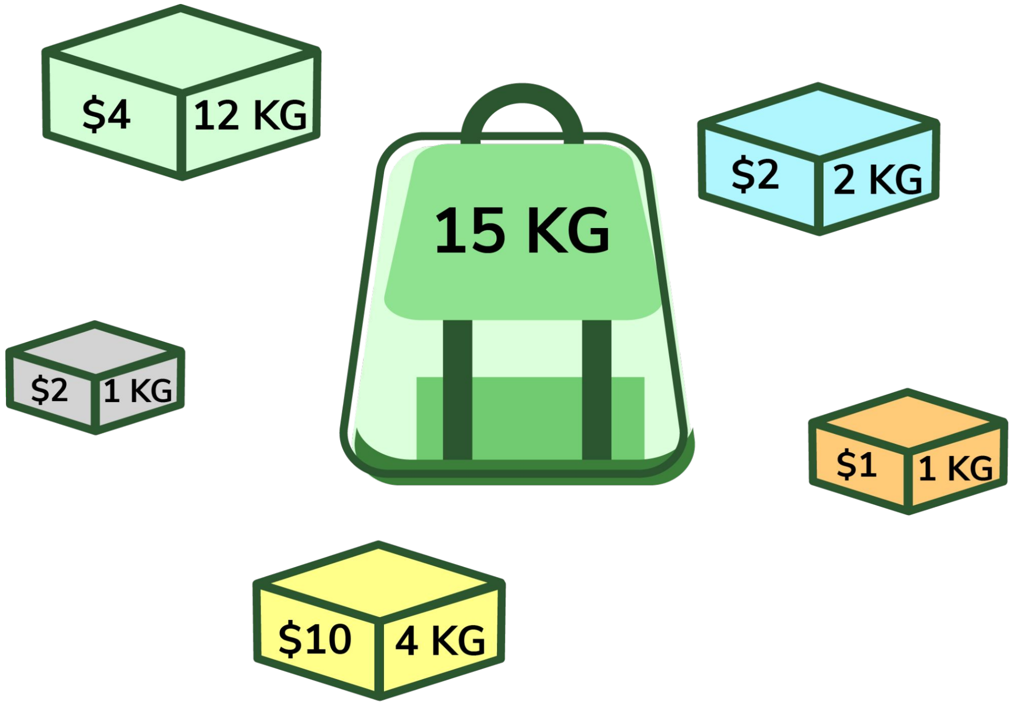
Model-based Algorithms for the 0-1 Time-Bomb Knapsack Problem

Stefano Chen
IN2000246

Introduction

0-1 Knapsack Problem:

- Classic problem in **combinatorial optimization**
- We have a fixed knapsack **capacity**
- A set of items
- Each item has a **weight** and a **profit**
- **Maximize the profits** without exceeding the capacity
- The problem is **NP-hard**



0-1 Time-Bomb Knapsack Problem

- Variant of the 01-KP
- Some items are time-bombs
- The objective is the maximization of the **expected profit**

A non-linear model

The 01-TB-KP can be fully defines by the following non-linear model

$$(NL) \quad \max \left(\sum_{k \in N} p_k x_k \right) \left(\prod_{k \in K} q_k x_k \right) \tag{2}$$

$$s.t. \quad \sum_{k \in N} w_k x_k \leq c \tag{3}$$

$$x_i \in \{0, 1\} \qquad i \in N \tag{4}$$

Linearization

The proposed methods are iterative and based on the following property.

This property allows the solution of the 01-TB-KP as a sequence of integer linear programming models

$$\prod_{k \in K} \alpha_k = \beta \Longleftrightarrow \sum_{k \in K} \log(\alpha_k) = \log(\beta)$$

Optimization of Profits

BestHeuValue: current best solution value
LastExactProfit: objective function value of the last problem

$$\left(P^{(j)}\right) \quad \max \sum_{k \in N} p_k x_k \tag{6}$$

$$s.t. \quad \sum_{k \in N} w_k x_k \leq c \tag{7}$$

$$\sum_{k \in N} \log(q_k) x_k \geq \log\left(\frac{BestHeuValue}{LastExactProfit}\right) \tag{8}$$

$$\sum_{k \in N : \hat{x}_k^{(i)} = 1} x_k + \sum_{k \in N : \hat{x}_k^{(i)} = 0} (1 - x_k) \leq n - 1 \qquad i = 1, 2, \dots, j-1 \tag{9}$$

$$x_i \in \{0, 1\} \qquad i \in N \tag{10}$$

Optimization of Probabilities

BestHeuValue: current best solution value
LastExactProbability: survival probability of the last solution

$$\left(S^{(j)}\right) \quad \max \sum_{k \in N} \log(q_k) x_k \tag{11}$$

$$s.t. \quad \sum_{k \in N} w_k x_k \leq c \tag{12}$$

$$\sum_{k \in N} p_k x_k \geq \frac{BestHeuValue}{LastExactProbability} \tag{13}$$

$$\sum_{k \in N : \hat{x}_k^{(i)} = 1} x_k + \sum_{k \in N : \hat{x}_k^{(i)} = 0} (1 - x_k) \leq n - 1 \qquad i = 1, 2, \dots, j-1 \tag{14}$$

$$x_i \in \{0; 1\} \qquad i \in N \tag{15}$$

IterativeAlgorithmP

IterativeAlgorithmS

Algorithm 1: IterativeAlgorithmP

Data: A 01-TB-KP instance

Result: A solution and an upper bound for the optimal solution value

BestHeuSol=∅;

BestHeuValue=0;

LastExactProfit=$+\infty$;

$j = 0$;

Build model $P^{(0)}$;

while *BestHeuValue<LastExactProfit and computational time* $< MT$ **do**

  Solve model $P^{(j)}$ with a maximum computation time of ML seconds;

  if $P^{(j)}$ *has been solved to optimality* **then**

    if $P^{(j)}$ *is feasible* **then**

      LastExactProfit=Value of the optimal solution of $P^{(j)}$;

    else

      LastExactProfit=BestHeuValue;

    end

  end

  if *BestHeuValue<LastExactProfit* **then**

    $\hat{x}^{(j)}$= Retrieved solution of $P^{(j)}$;

    if *Eq.* (2) *evaluated on* $\hat{x}^{(j)}$ *>BestHeuValue* **then**

      BestHeuSol=$\hat{x}^{(j)}$;

      BestHeuValue=Value($\hat{x}^{(j)}$);

    end

    $j = j + 1$;

    Build model $P^{(j)}$;

  end

end

return BestHeuSol, LastExactProfit;

Algorithm 2: IterativeAlgorithmS

Data: A 01-TB-KP instance

Result: A solution and a value indicating whether the solution is optimal or heuristic

BestHeuSol=∅;

BestHeuValue=0;

LastExactProbability=1;

$j = 0$;

Build model $S^{(0)}$;

while *Computational time* $< MT <$ *MaxTime* **do**

  Solve model $S^{(j)}$ with a maximum computation time of ML seconds;

  if $S^{(j)}$ *has been solved to optimality* **then**

    if $P^{(j)}$ *is feasible* **then**

      LastExactProbability=Value of the optimal solution of $S^{(j)}$;

    else

      return BestHeuSol,“exact”;

    end

  end

  $\hat{x}^{(j)}$=Optimal solution of $S^{(j)}$;

  $\hat{y}^{(j)}$=Optimal solution of the deterministic Knapsack problem with non-bomb unselected items and residual capacity;

  $\hat{z}^{(j)} = \hat{x}^{(j)} \vee \hat{y}^{(j)}$;

  if *Eq.* (2) *evaluated on* $\hat{z}^{(j)}$ *>BestHeuValue* **then**

    BestHeuSol=$\hat{z}^{(j)}$;

    BestHeuValue=Value($\hat{z}^{(j)}$);

  end

  $j = j + 1$;

  Build model $S^{(j)}$;

end

return BestHeuSol,“heuristic”;

Scalability Analysis

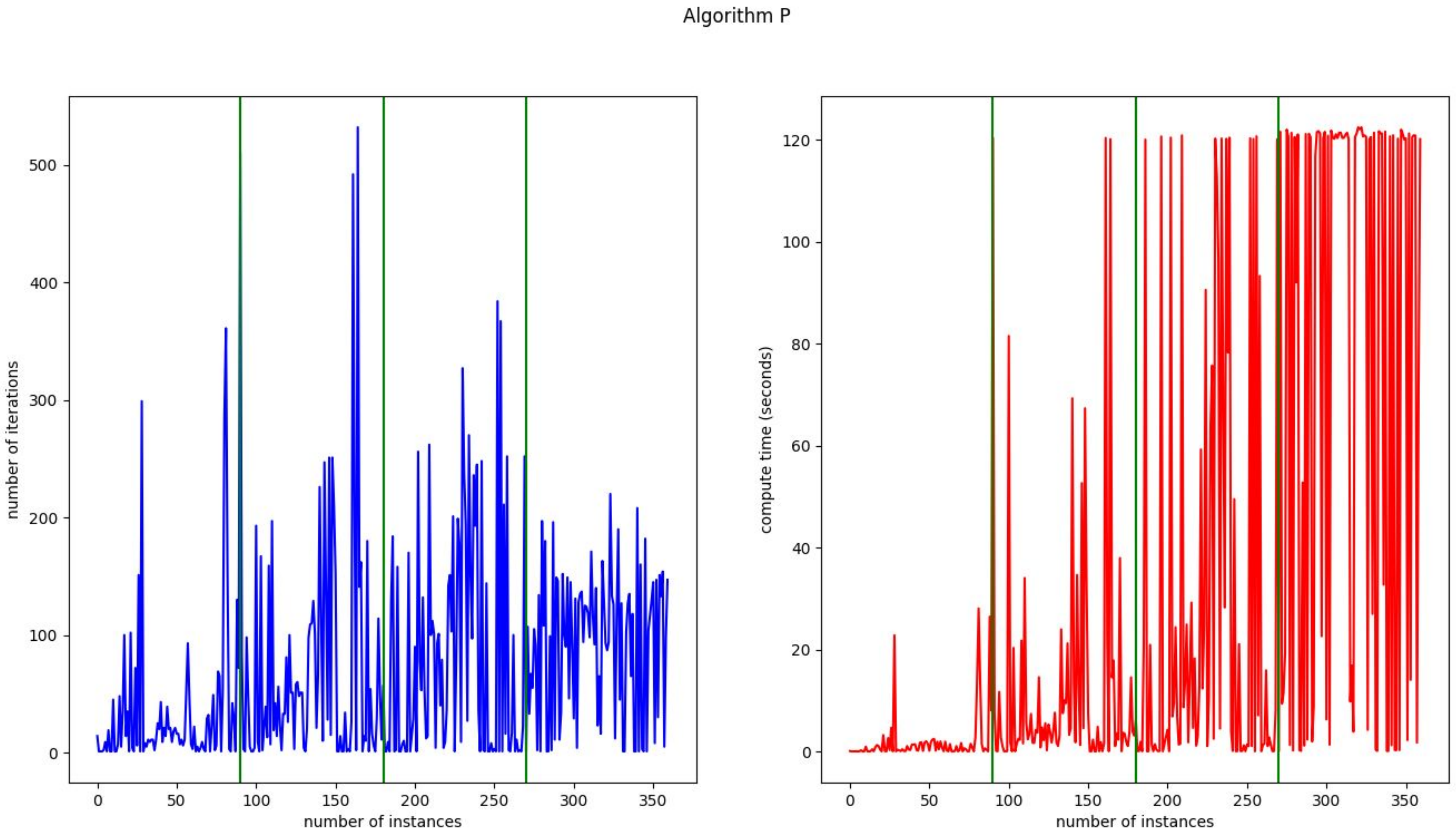
Experiment steps

- Python 3.12.6
- Gurobi 12.0.1
- Testing on small instance (size 4)
- Scalability analysis
  - MT=120 seconds
  - ML = 1 second
  - 360 instances
  - instances of increasing size [100, 500, 1000, 5000]
  - generated by alberto-santini on GitHub

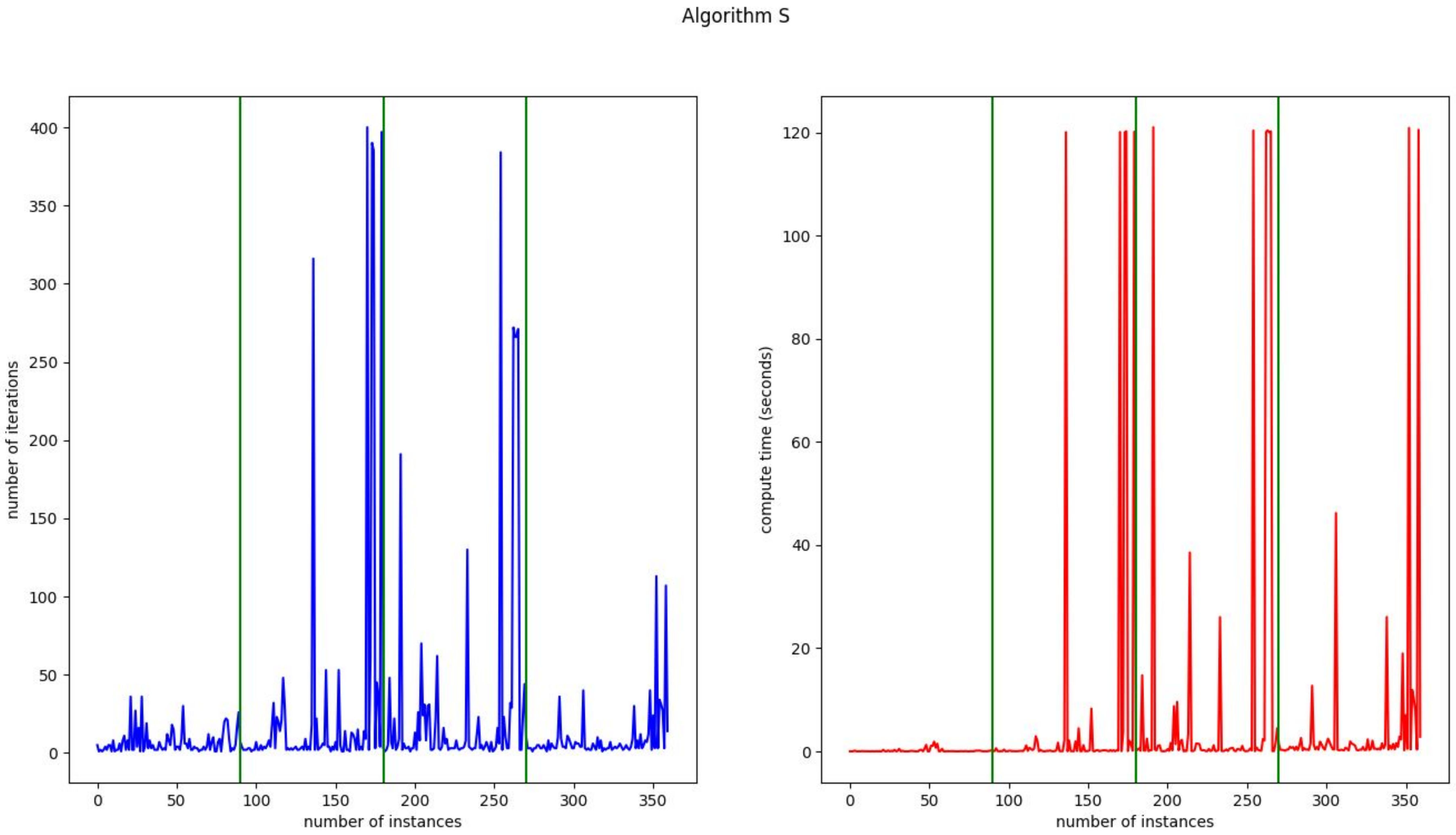
Testing Instance

```
Testing Instance
        number of items : 4
        knapsack's capacity : 5
        items weights : [2, 3, 1, 4]
        items profits : [10, 20, 15, 40]
        items survival probabilities : [0.9, 1, 0.8, 1]

IterativeAlgorithmP
        solution : [0.0, 0.0, 1.0, 1.0]
        expected profit : 44.0
        upper bound : 55.0
        number of iterations : 1
        computational time : 0.025 seconds

IterativeAlgorithmS
        solution : [0, 0, 1.0, 1.0]
        expected profit : 44.0
        solution type : exact
        number of iterations : 3
        computational time : 0.007 seconds
```

Scalability Algorithm P



Scalability Algorithm S



Final Comparison

		Size 100	Size 500	Size 1000	Size 5000	Total
Algorithm P	Avg. N iterations	33.58	75.06	83.72	91.81	71.04
	Avg. Comp time (seconds)	1.90	12.89	29.22	78.57	30.64
Algorithm S	Avg. N iterations	6.88	29.63	29.07	9.47	18.76
	Avg. Comp time (seconds)	0.15	7.17	9.62	4.99	5.48

Thank You

for your attention