Increasing profit from fees of cryptocurrencies

Besart Dollma Noa Oved
Advisor: Itay Tsabary

Technion - Israel Institute of Technology

October, 2018

Cryptocurrencies
The knapsack and dependency knapsack problems
Solutions
Implementation
Results
Conclusions

Outline

1 Cryptocurrencies

2 The knapsack and dependency knapsack problems

3 Solutions

4 Implementation

5 Results

6 Conclusions

Outline

1. Cryptocurrencies
2. The knapsack and dependency knapsack problems
3. Solutions
4. Implementation
5. Results
6. Conclusions

Cryptocurrencies

- A cryptocurrency is digital asset designed to work as a medium of exchange that uses strong cryptography to secure financial transactions, control the creation of additional units, and verify the transfer of assets.

- We will focus on Bitcoin.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies
Cryptocurrencies
Blockchain
Transactions
MemPool
Miners

The knapsack and dependency knapsack problems

Solutions

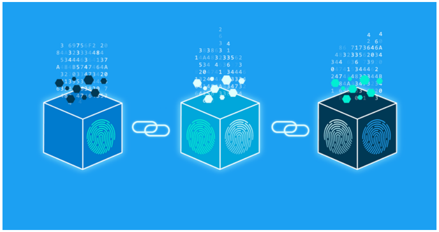
Implementation

Results

Conclusions

Blockchain

- The validity of each cryptocurrency's coins is provided by a blockchain.
- Each block typically contains a hash pointer as a link to a previous block, a timestamp and transaction data.



Increasing profit from fees of cryptocurrencies

Cryptocurrencies
- Cryptocurrencies
- Blockchain
- Transactions
- MemPool
- Miners

The knapsack and dependency knapsack problems

Solutions

Implementation

Results

Conclusions

Blockchain

- Resistant to data modification.
- On average, every 10 minutes a new block is added to the blockchain.
- Block size is 1 MB.

Transactions

- When transferring cryptocurrency from one party to another, a transaction is created and added to the Mempool (explained later).
- Among other things, transactions contain:

ID	672e2c74d410d0a5b689925155098c9a39
Fee	0.00015820 BTC
Size	224 bytes
Depends	[ ]

MemPool

- Mempool (‘Memory’+‘Pool’) is a pool of memorized, held data.
- The data that is being stored on the Mempool are unconfirmed transactions that are currently stuck on the network.
- We will denote the size of the mempool by n.
- $n \approx 16000$.

Cryptocurrencies
Cryptocurrencies
Blockchain
Transactions
MemPool
Miners
The knapsack and dependency knapsack problems
Solutions
Implementation
Results
Conclusions

Miners

- In cryptocurrency networks, mining is a validation of transactions.
- For this effort, successful miners obtain new cryptocurrency as a reward.
- For each transcation the miner includes in a block, he collects its fee.
- $\Rightarrow$ The miner's motivation is to maximize the sum of the fees of the transactions that he includes in the block. This needs to be done under the size and dependency constraints.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies
Cryptocurrencies
Blockchain
Transactions
MemPool
Miners

The knapsack and dependency knapsack problems

Solutions

Implementation

Results

Conclusions

Outline

1 Cryptocurrencies

2 The knapsack and dependency knapsack problems

3 Solutions

4 Implementation

5 Results

6 Conclusions

The knapsack problem

Definition

We are given a knapsack (block) of size W and n transactions $\{a_1, a_2, ..., a_n\}$.
Each transaction a_i has size $s_i > 0$ and fee $f_i \geq 0$.
We are to find $I \subseteq [n]$ such that:

$$I = \arg\max_{J \subseteq [n]} \{\sum_{j \in J} f_j\}$$

such that

$$\sum_{j \in J} s_j \leq W$$

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

The knapsack problem

The dependency knapsack problem

Solutions

Implementation

Results

Conclusions

The dependency knapsack problem

Definition

We are given a knapsack (block) of size W and n transactions $\{a_1, a_2, ..., a_n\}$ as mentioned in the knapsack problem.
Each transaction a_i has in addition to its size and fee also a set of transaction which it depends upon.

The dependency knapsack problem

Definition cont.

The goal is to find $\bar{V} \subseteq V$ such that:

$$\bar{V} = \arg\max_{J \subseteq V} \{ \sum_{v \in J} f(v) \}$$

such that

$$\sum_{v \in J} s(v) \leq W$$

and the dependency constraints are preserved $\forall v \in \bar{V}$.

The dependency knapsack problem

- No circular dependencies exist
- We will treat the input as a directed acyclic graph $G = (V, E)$.
- Each node v represents a transaction that has fee $f(v)$ and size $s(v)$.
- Each edge (i, j) represents that transaction j is dependent upon transaction i.
- Transaction j can be selected if transaction m is selected $\forall (m, j) \in E$.

The dependency knapsack problem

Example

- The knapsack size is $W = 11$.

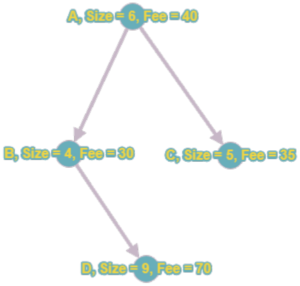


Figure: Example of Knapsack with dependencies

- Every transaction is dependent upon A, hence A must be selected.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

The knapsack problem

The dependency knapsack problem

Solutions

Implementation

Results

Conclusions

The dependency knapsack problem

Example

- The knapsack size is $W = 11$.

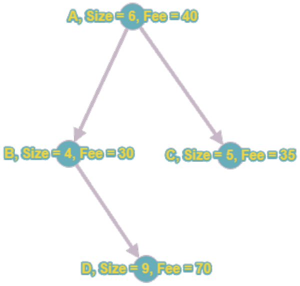


Figure: Example of Knapsack with dependencies

- Every transaction is dependent upon A, hence A must be selected. Optimal solution is $\{A, C\}$.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

The knapsack problem

The dependency knapsack problem

Solutions

Implementation

Results

Conclusions

NP-Completeness

- The decision problem form of the knapsack problem is NP-complete.
- The dependency knapsack problem is a generalization of the knapsack problem $\Rightarrow$ also NP-Complete.
- Thus there is no known algorithm both correct and polynomial for any of the problems.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

The knapsack problem

The dependency knapsack problem

Solutions

Implementation

Results

Conclusions

Outline

1 Cryptocurrencies

2 The knapsack and dependency knapsack problems

3 Solutions

4 Implementation

5 Results

6 Conclusions

Knapsack solvers

Exhaustive Search

- Trivial approach.
- Check all the subsets $J \subseteq [n]$.
- Optimal solution.
- Runtime is $\mathcal{O}(2^n)$, hence exponential.
- We remind that $n \approx 16000$, therefore infeasible.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Knapsack solvers

Exhaustive Search

Dynamic Programming

Greedy approximation

$(1+\varepsilon)$ approximation

Dependency knapsack solvers

Incremental greedy approximation

Implementation

Results

Conclusions

Knapsack solvers
Dynamic Programming

- Assume $s_1, s_2, \dots, s_n, W \in \mathbb{N}$. It holds on real data.
- For each $k \in \{0, 1, \dots, n\}$ and for each $w \in \{0, 1, \dots, W\}$ define:
- $F(k, w) \triangleq$ the maximal profit when choosing from transactions $\{a_1, \dots, a_k\}$ and the size of the block is w.
- It holds:

$$F(k,w) = \begin{cases} 0 & k = 0 \text{ or } w = 0 \\ F(k-1, w) & w, k > 0 \text{ and } s_k > w \\ \xi & w, k > 0 \text{ and } s_k \leq w \end{cases}$$

$$\xi = \max\{F(k-1, w), f_k + F(k-1, w - s_k)\}$$

- Optimal solution.
- Runtime is $\mathcal{O}(nW)$, hence pseudo-polynomial.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions
- Knapsack solvers
  - Exhaustive Search
  - Dynamic Programming
  - Greedy approximation
  - $(1 + \varepsilon)$ approximation
- Dependency knapsack solvers
- Incremental greedy approximation

Implementation

Results

Conclusions

Knapsack solvers

Greedy approximation

- Sort the transactions by the $\frac{f_i}{s_i}$ ratio in descending order.
- Iterate in this order and add transactions to the block until the next transaction can't be added.
- Not optimal but a 2-approx.
- Runtime is $\mathcal{O}(n \log n)$.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions
- Knapsack solvers
  - Exhaustive Search
  - Dynamic Programming
  - Greedy approximation
  - $(1 + \varepsilon)$ approximation
- Dependency knapsack solvers
- Incremental greedy approximation

Implementation

Results

Conclusions

Knapsack solvers

$(1+\varepsilon)$ approximation

- Let $0 \leq \varepsilon \leq 1$.
- Denote by *greedySol* the value of the greedy approximation and let

$$a = \varepsilon \cdot greedySol$$

- Denote

$$V_a = \{a_i \mid f_i < a\}$$

$$V_a^C = \{a_i \mid f_i \geq a\}$$

- For each $J \subseteq V_a^C$ such that $|J| \leq \frac{2}{\varepsilon}$:
  - Run the greedy approximation on V_a with block size $W' = W - \sum_{j \in J} s_j$ and denote the solution by I_j.
- Output $I_j \cup J$ with maximal profit.

Knapsack solvers
$(1+\varepsilon)$ approximation

- Not optimal but a $(1+\varepsilon)$ approx.
- Runtime is $\mathcal{O}(n^{1+\frac{2}{\varepsilon}} \cdot \log n)$.
- If $\varepsilon \to 0$ we receive the exhaustive search. Indeed it holds that $\frac{2}{\varepsilon} \to \infty$ (non-polynomial).
- If $\varepsilon \to 1$ we receive the greedy approximation. However the runtime is longer.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions
- Knapsack solvers
  - Exhaustive Search
  - Dynamic Programming
  - Greedy approximation
  - $(1+\varepsilon)$ approximation
- Dependency knapsack solvers
- Incremental greedy approximation

Implementation

Results

Conclusions

Solutions for the dependency knapsack problem

- We will discuss only the two approximations.
- The idea behind the algorithms remains the same.
- The algorithms are adjusted to supply the dependency constraints.

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions
Knapsack solvers
Dependency knapsack solvers
Greedy approximation
$(1 + \varepsilon)$ approximation
Incremental greedy approximation

Implementation

Results

Conclusions

Solutions for the dependency knapsack problem

Definition

For each transaction v (node in the graph G)

$$Ancestor(v) \triangleq \{j \mid \text{there exists a path from } j \text{ to } v \text{ in } G\}$$

Note

Pay attention that $v \in Ancestor(v)$.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions
- Knapsack solvers
- Dependency knapsack solvers
  - Greedy approximation
  - $(1+\varepsilon)$ approximation
- Incremental greedy approximation

Implementation

Results

Conclusions

Dependency knapsack solvers

Greedy approximation

Algorithm

1. Calculate the sets $Ancestor(v)$ for all $v \in V$.
2. Pick $Ancestor(v)$ with the maximal

$$\frac{\sum_{j \in Ancestor(v)} f(j)}{\sum_{j \in Ancestor(v)} s(j)}$$

   ratio and add the transactions of the set to the knapsack.
3. Remove the transactions we just added to the knapsack from other sets and continue from 2 until we can't fit anything in the block.

- Runtime is $\mathcal{O}(n^3)$.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions
- Knapsack solvers
- Dependency knapsack solvers
  - Greedy approximation
  - $(1+\varepsilon)$ approximation
- Incremental greedy approximation

Implementation

Results

Conclusions

Dependency knapsack solvers
Greedy approximation

Example

- The knapsack size is $W = 11$.

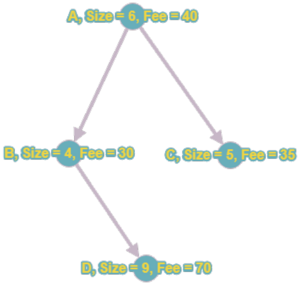


Figure: Example of Knapsack with dependencies

- We remind that the optimal solution is $\{A, C\}$.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions
- Knapsack solvers
- Dependency knapsack solvers
  - Greedy approximation
  - $(1 + \varepsilon)$ approximation
- Incremental greedy approximation

Implementation

Results

Conclusions

Dependency knapsack solvers
Greedy approximation

Example

- $Ancestor(D) = \{D, B, A\}$ and $Ratio(D) = \frac{140}{19} \approx 7.36$.
- $Ancestor(B) = \{B, A\}$ and $Ratio(B) = \frac{70}{10} = 7$.
- $Ancestor(C) = \{C, A\}$ and $Ratio(C) = \frac{75}{11} \approx 6.81$.
- $Ancestor(A) = \{A\}$ and $Ratio(A) = \frac{40}{6} \approx 6.66$.
- The greedy approximation will try to add $Ancestor(D)$ to the knapsack but won't succeed since the size of the set is 19 and the knapsack size is 11.
- The algorithm will then add $Ancestor(B)$ to the solution and remove $\{B, A\}$ from the other sets.
- The algorithm will output $\{A, B\}$. Output value is 70.
- Not optimal.

Dependency knapsack solvers

$(1+\varepsilon)$ approximation

- Similar to the $(1+\varepsilon)$ approximation for the knapsack problem.
- Adjusted to the dependency knapsack problem using $\mathit{Ancestor}(\cdot)$ sets.
- Runtime is $\mathcal{O}(n^{3+\frac{2}{\varepsilon}})$.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Knapsack solvers

Dependency knapsack solvers

Greedy approximation

$(1+\varepsilon)$ approximation

Incremental greedy approximation

Implementation

Results

Conclusions

Dependency knapsack solvers

$(1+\varepsilon)$ approximation

Example

- Let $\varepsilon = 0.1$
- $a = \varepsilon \cdot \mathit{greedySol} = 7$
- $$V_a^C = \{a_i \mid f_i \geq a\} = \{A, B, C, D\}$$
- Therefore we will check all $J \subseteq V_a^C$ such that $|J| \leq \frac{2}{\varepsilon} = 20$, in particular $\{A, C\}$.
- In this example the algorithm outputs the optimal solution.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Knapsack solvers

Dependency knapsack solvers

Greedy approximation

$(1+\varepsilon)$ approximation

Incremental greedy approximation

Implementation

Results

Conclusions

An incremental solution to the greedy approximation

- Suppose at time t we have the solution of the dependency knapsack problem.
- Now suppose that at time $t+1$ some transactions were added to the mempool but no transactions were removed.
- Suppose also that the added transactions may be dependent on transactions from time t but not vice verse.

Can we solve the dependency knapsack at time $t+1$ using the solution at time t and reduce the runtime?

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Knapsack solvers

Dependency knapsack solvers

Incremental greedy approximation

Implementation

Results

Conclusions

An incremental solution to the greedy approximation

Goal:

Improve the running time.

- $Ancestor(v)$ remains identical for all the transactions which were also at time t.
- The solution value can't improve the solution value of the greedy approximation.
- In most cases we got the same solution value.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Knapsack solvers

Dependency knapsack solvers

Incremental greedy approximation

Implementation

Results

Conclusions

An incremental solution to the greedy approximation

Algorithm

1. Calculate *Ancestor*(v) for all transactions v.
2. Denote by

$$\alpha = \max_{v} \frac{\sum_{j \in Ancestor(v)} f(j)}{\sum_{j \in Ancestor(v)} s(j)}$$

   the maximal ratio of fee over size of the transactions that weren't in the previous solution.
3. Add to the solution all the *Ancestor*$(\cdot)$ sets from the previous solution with a ratio bigger than α and remove them from the mempool. Use the greedy approximation algorithm on the transactions left on the mempool with the new size of the block.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions
- Knapsack solvers
- Dependency knapsack solvers
- Incremental greedy approximation

Implementation

Results

Conclusions

Outline

1 Cryptocurrencies

2 The knapsack and dependency knapsack problems

3 Solutions

4 Implementation

5 Results

6 Conclusions

Implementation

- We implemented the algorithms using Python.
- Graphs aren't build in, hence we used the networkx package.
- Everything can be found in our github.
- Caching was disabled through out the tests.

Cryptocurrencies
The knapsack and dependency knapsack problems
Solutions
Implementation
Results
Conclusions

Outline

1. Cryptocurrencies
2. The knapsack and dependency knapsack problems
3. Solutions
4. Implementation
5. Results
6. Conclusions

The knapsack problem

- Exhaustive search and dynamic programming are infeasible
  1. For $n = 25$ the exhaustive search runs on average 8 sec.
  2. Dynamic programming for $n = 1000$:
     - If $W = 1000$ runs on average 1 sec.
     - If $W = 1$ MB (block size) runs on average 25 sec.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Implementation

Results

The knapsack problem

The dependency knapsack problem

Real data

The incremental solution

Conclusions

The knapsack problem

- Picking the right ε for the $(1+\varepsilon)$ approximation
  1. Right ε means a better solution of the $(1+\varepsilon)$ approximation than the greedy approximation.
  2. The right ε is usually very small. This makes $\frac{2}{\varepsilon}$ very big $\Rightarrow$ long runtime (exponential on $\frac{2}{\varepsilon}$).
  3. Hence, if $|V_a^C| > 20$ we reduce it to 20 it by moving transactions to V_a.
  4. $\Rightarrow$ The algorithm isn't a $(1+\varepsilon)$ approx.
  5. We studied different ways to pick which transactions to leave in V_a^C.

The dependency knapsack problem

Experiment

- How do dependencies influence the runtime of the algorithms?
- The greedy approximation for the knapsack runs $\mathcal{O}(n \cdot \log n)$.
- The greedy approximation for the dependency knapsack runs $\mathcal{O}(n^3)$.

Experiment

For each $n = 1, 11, 21, ..., 141$ we generated each time n random transactions a_i such that $1 \leq s_i, f_i \leq 100$. $W = 10000$. The dependencies are also generated randomly such that there are no circular dependencies. The result's data points are calculated as the average of 10 runs.

Cryptocurrencies
The knapsack and dependency knapsack problems
Solutions
Implementation
Results
The knapsack problem
The dependency knapsack problem
Real data
The incremental solution
Conclusions

The dependency knapsack problem

Experiment

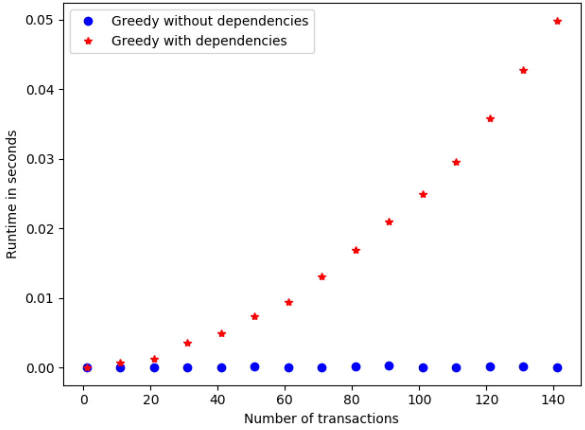


Figure: Runtime in seconds as a function of n

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Implementation

Results

- The knapsack problem
- The dependency knapsack problem
- Real data
- The incremental solution

Conclusions

Real data

- We checked the dependency knapsack approximations that we implemented on real data.
- The data was sampled using the Bitcoin API.
- Every day, the first sample $t = 1$ returns the whole mempool.
- Other samples $t > 1$ through out the day return two sections:
    - An added section that contains the transactions that are added in the mempool after the last sample at $t - 1$.
    - The added transactions can be dependent upon transactions that were in the mempool at time $t - 1$ but not vice verse.
    - A removed section that contains the transactions that are removed from the mempool after the last sample at $t - 1$.
    - The removed section is usually empty.

Cryptocurrencies
The knapsack and dependency knapsack problems
Solutions
Implementation
Results
The knapsack problem
The dependency knapsack problem
Real data
The incremental solution
Conclusions

Real Data

- For $\varepsilon > 0.03$ the two approximations yield the same solution almost every day because $V_a^C = \emptyset$.
- We explore the greedy approximation and $(1 + \varepsilon)$ approximation for $\varepsilon = 0.03$ through out different days.
- The data was sampled at different days at time sample $t = 1$.
- Furthermore, if $|V_a^C| > 10$ it is reduced to 10 using the fee criterion.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Implementation

Results

The knapsack problem

The dependency knapsack problem

Real data

The incremental solution

Conclusions

Real Data

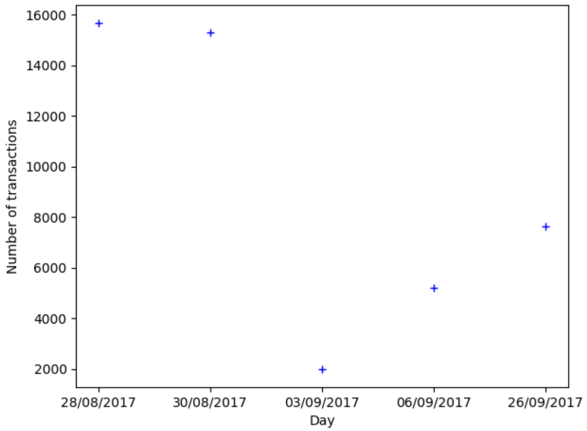


Figure: n at $t = 1$ per day

Cryptocurrencies
The knapsack and dependency knapsack problems
Solutions
Implementation
Results
The knapsack problem
The dependency knapsack problem
Real data
The incremental solution
Conclusions

Real Data

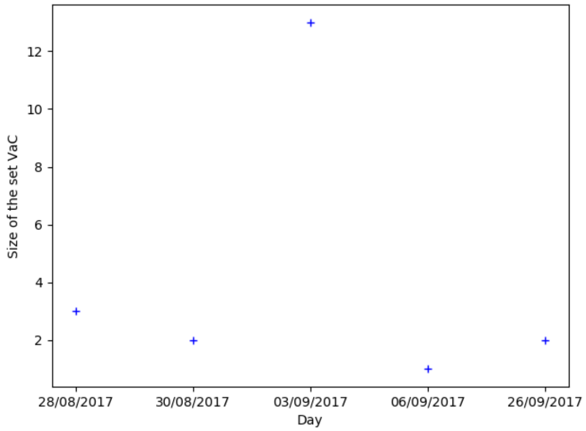


Figure: $|V_a^C|$ per day when $\varepsilon = 0.03$

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Implementation

Results

- The knapsack problem
- The dependency knapsack problem
- Real data
- The incremental solution

Conclusions

Real Data

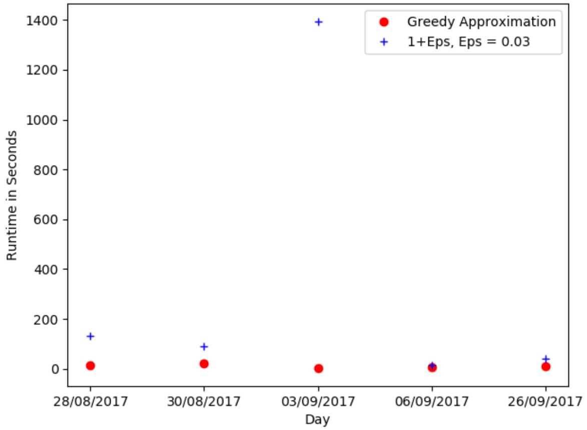


Figure: Runtime in seconds per day

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Implementation

Results
- The knapsack problem
- The dependency knapsack problem
- Real data
- The incremental solution

Conclusions

Real Data

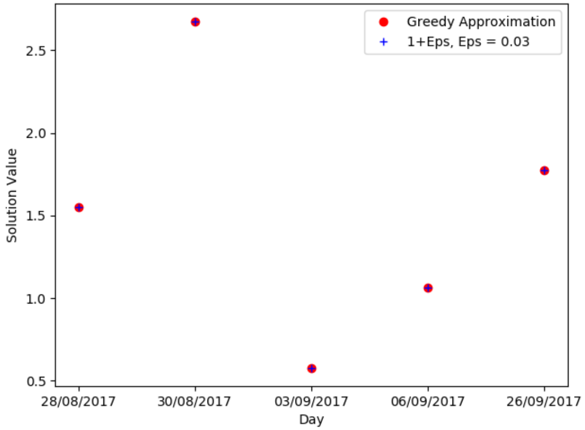


Figure: The solution value per day

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Implementation

Results

The knapsack problem

The dependency knapsack problem

Real data

The incremental solution

Conclusions

The incremental solution

Real Data

- The real data provides the constraints needed in order to use the incremental solution, since most of the time there are no removed transactions but only added ones.

Experiment

This data was sampled on 28/08/2017.

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Implementation

Results
- The knapsack problem
- The dependency knapsack problem
- Real data
- The incremental solution
  - Real Data

Conclusions

The incremental solution

Real Data

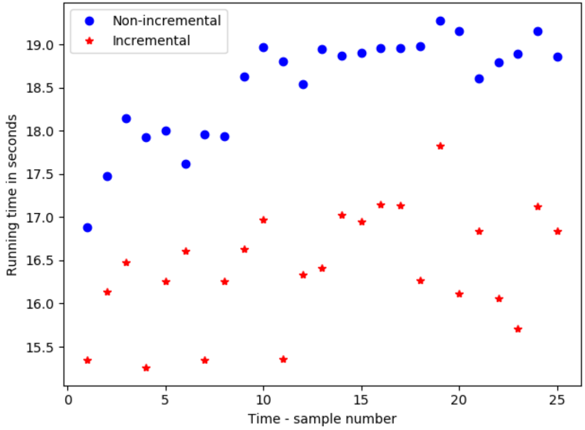


Figure: Runtime in seconds as a function of t

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Implementation

Results
- The knapsack problem
- The dependency knapsack problem
- Real data
- The incremental solution
  - Real Data

Conclusions

The incremental solution

Real Data

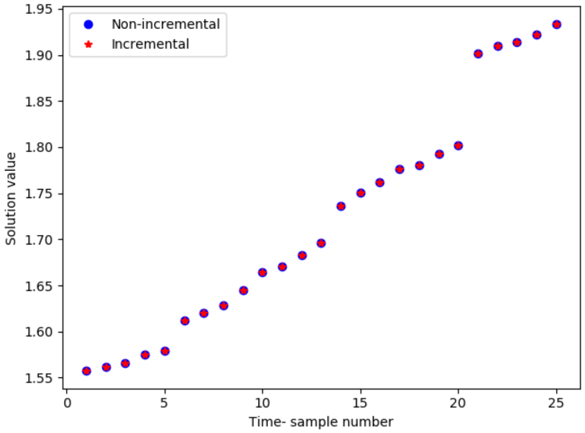


Figure: Solution value as a function of t

Increasing profit from fees of cryptocurrencies

Cryptocurrencies

The knapsack and dependency knapsack problems

Solutions

Implementation

Results

The knapsack problem

The dependency knapsack problem

Real data

The incremental solution

Real Data

Conclusions

Outline

1 Cryptocurrencies

2 The knapsack and dependency knapsack problems

3 Solutions

4 Implementation

5 Results

6 Conclusions

Conclusions

- The exhaustive search algorithm and the dynamic programming algorithms are not feasible for big n and big W.
- The dependency knapsack problem is harder to solve than the knapsack problem as we saw.

Conclusions

- The $(1+\varepsilon)$ approximation is not feasible for the dependency knapsack problem.
- Reason: the greedy approximation value $\gg$ fees of transactions $\Rightarrow \varepsilon \ll 1$ which means that $\frac{2}{\varepsilon}$ is big
- $\Rightarrow$ The algorithm becomes infesiable in terms of time and memory.
- On real data, the incremental solution algorithm gives the same results as the greedy approximation algorithm but it is faster most of the time.