Knapsack Problem

Paul Franklin, Yangqi Su, Yaying Shi

Presentation of the problem (v2)

Input

$S = \{1, 2, \dots, n\}$ having cost s_i

$V = \{v_1, v_2, \dots, v_n\}$ value of i of n

C max cost constraint

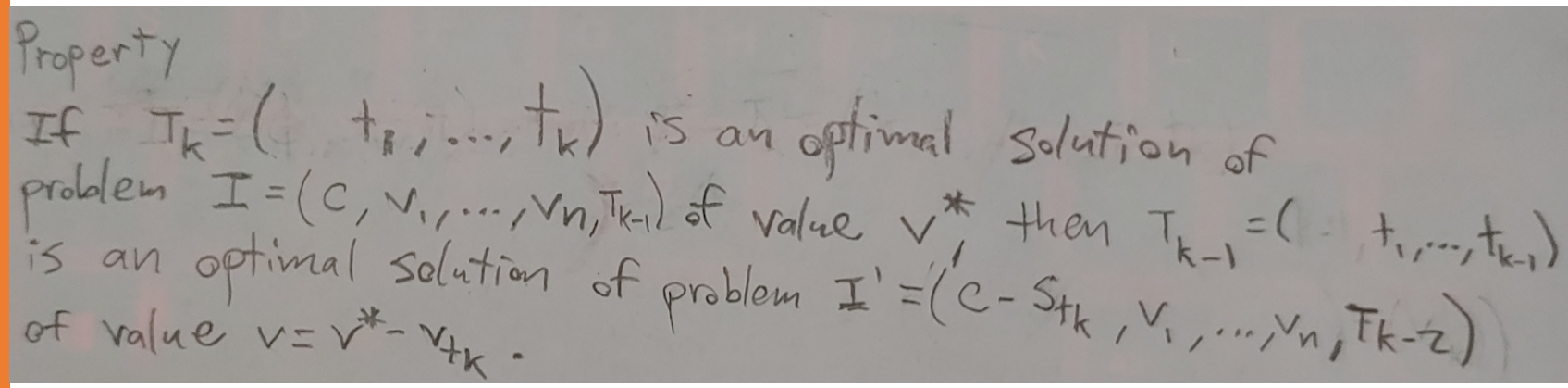
Output

$$T \subseteq S \ni \sum_{i \in T} s_i \leq C$$

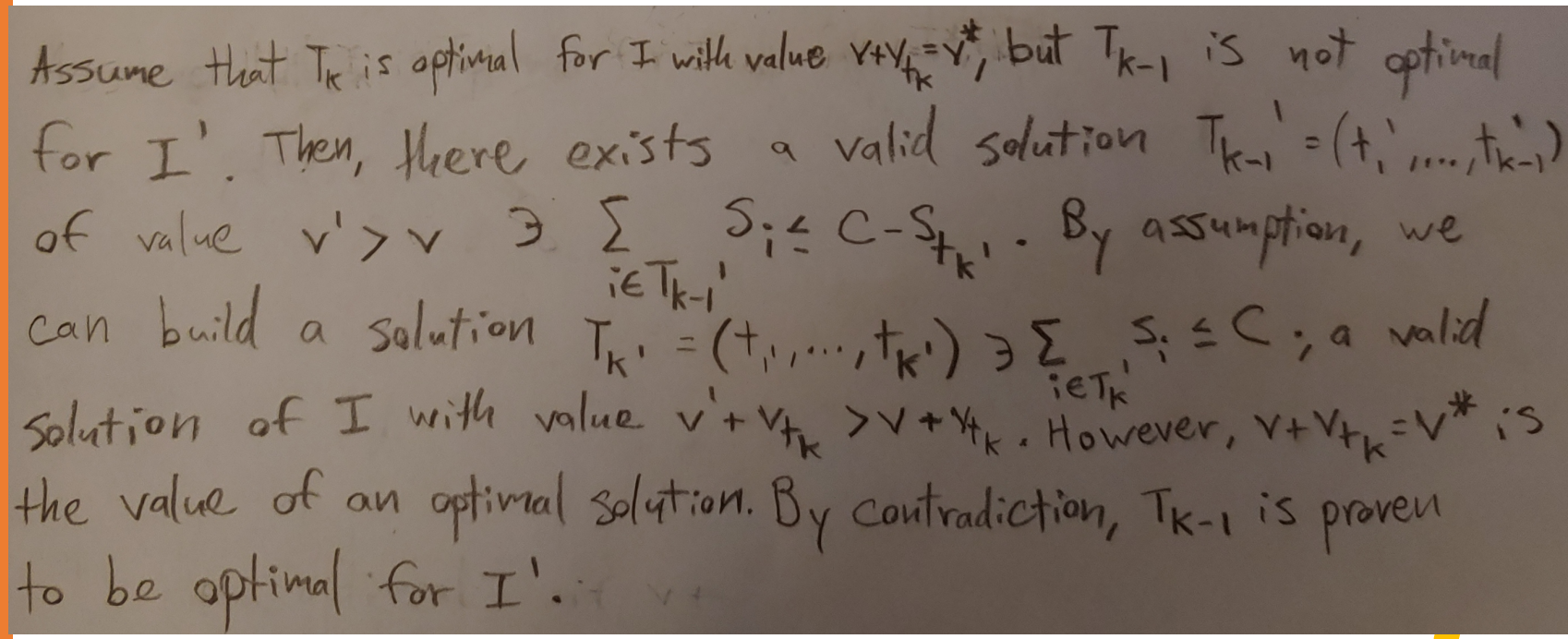
Metric

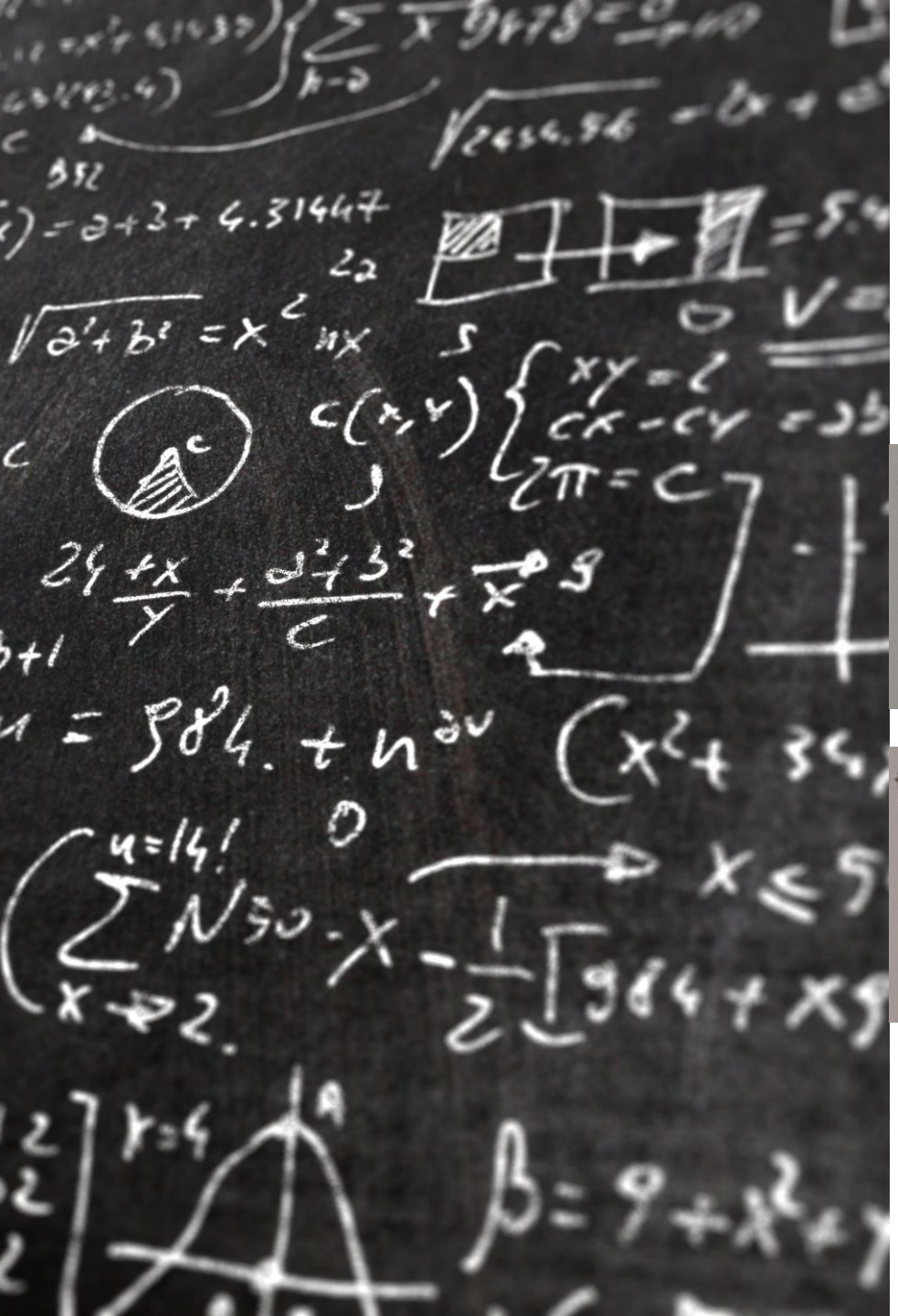
$$G = \max\left(\sum_{i \in T} v_i\right)$$

Property (v2)


Property
If $T_k = (t_1, \ldots, t_k)$ is an optimal solution of problem $I = (c, v_1, \ldots, v_n, T_{k-1})$ of value v^* then $T_{k-1} = (t_1, \ldots, t_{k-1})$ is an optimal solution of problem $I' = (c - s_{t_k}, v_1, \ldots, v_n, T_{k-2})$ of value $v = v^* - v_{t_k}$.

Proof by contradiction (v2)





Formula (v2)

Formula

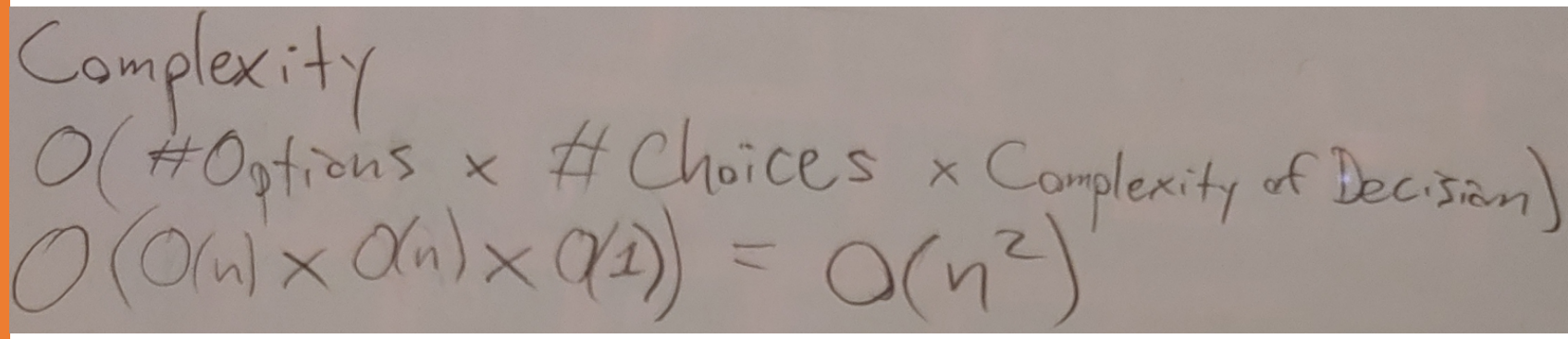
$$KS(C, T_k) = \max_{1 \le i \le n,\ \exists i \notin T_k} \left(KS(C - S_i, T_k \cup \{i\}) + V_i \right)$$

Formula (Continued)

$KS(0, T_k) = 0$ Base Case

$KS(C, T_k \ni \text{for every } i \in S, i \in T_k) = 0$ Base Case

Complexity (v2)



Presentation of the problem

- Input:

  For any n object set, we have size set S and value set V for those objects.

  S = $\{s_i | \, where \; i \; from \; 1 \; to \; n\} = \{s_1, s_2, s_3, \dots, s_n\}$

  V = $\{v_i | \, where \; i \; from \; 1 \; to \; n\} = \{v_1, v_2, v_3, \dots, v_n\}$

  You have a capacity of C which is a size limit.

- Output:

  A subset of S

  T = $\{s_i | \, where \; i \in T\}$

- Metric

  Max$\sum_{i \in T} v_i$

Important of this problem

- Any arrangement and schedule problem with a threshold
  1. If you have an exam tomorrow, how could you select the most value part in limited time.
  2. How could you schedule your travel plan with limited money?
  3. Data download management
  4. Cache select

Property first insight

If S = (k, $s_1, \dots, s_k$) is an optimal solution of problem I = (C, , $s_1, \dots, s_n$) of v^*, then S = (k-1, $s_1, \dots, s_{k-1}$) is optimal solution of problem $I' = ((C - s_k), s_1, \dots, s_{k-1}, s_{k+1}, \dots, s_n)$ of value v= v^* - v_i.

Property update

If S^* = (k, $s_1, \dots, s_k$) is an optimal solution of problem I = (C , $s_1 , \dots, s_n$) of v^*, then S^* is an optimal solution of problem $I' =$ (C, $s_1, \dots, s_{n-1}$) of value v= v* if $s_n \neq s_k$, S = (k-1, $s_1, \dots, s_{k-1}$) is an optimal solution of problem $I'' = ((C - s_n), s_1, \dots, s_{n-1})$ of value v= v^* - v_n if $s_n = s_k$

S^* = (k, $s_1, \dots, s_k$) I = (C , $s_1 , \dots, s_n$) $I' = (C, s_1, \dots, s_{n-1})$ $I'' = ((C - s_n), s_1, \dots, s_{n-1})$ S = (k-1, $s_1, \dots, s_{k-1}$) v= v^* - v_n

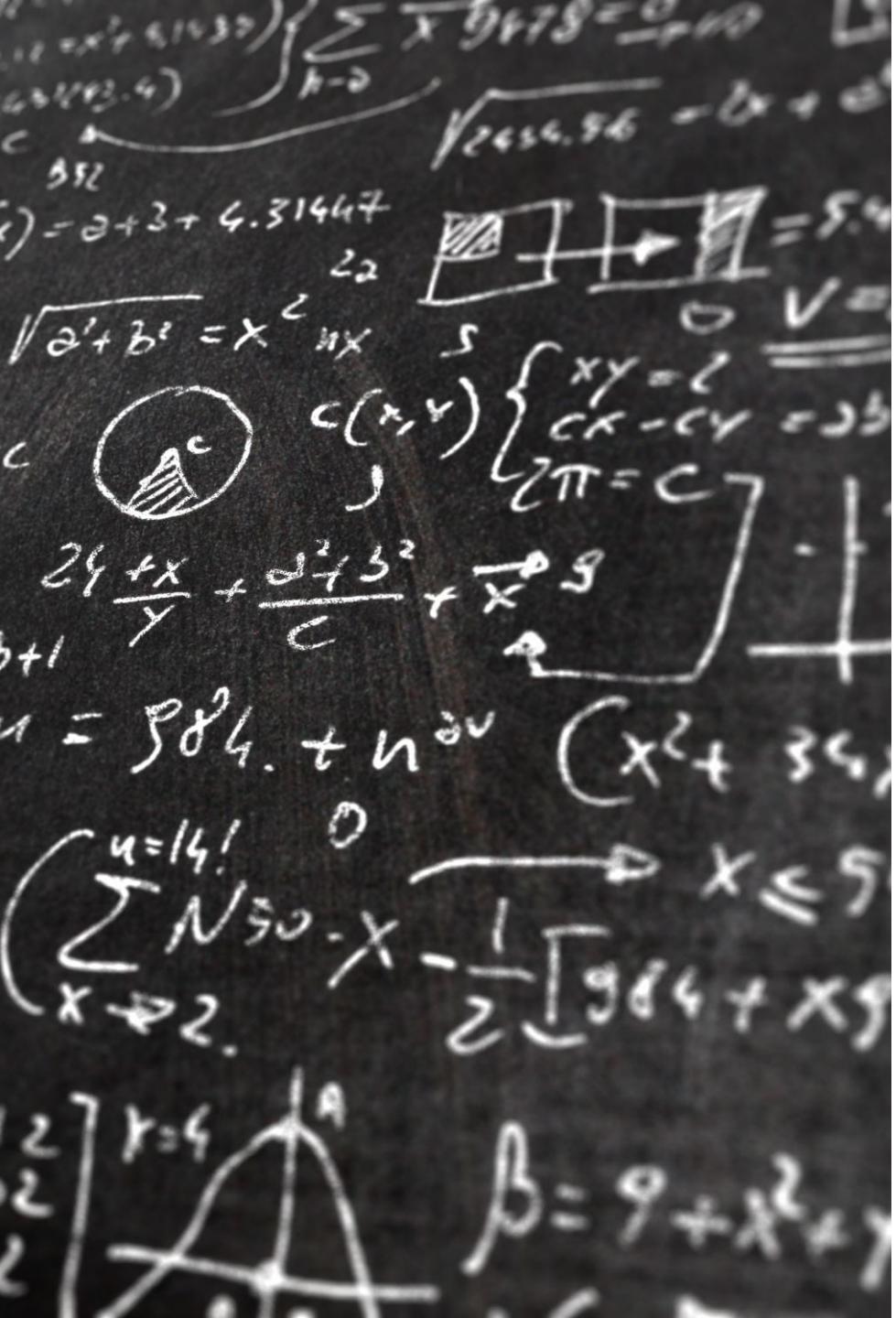
Proof by contradiction

Assume S^* is optimums for I but S^* is not optimums for I' and S is not optimums for I'',

(1)then there exist a valid solution S' for I' of $v' > v^*$. It's conflict with S^* is optimums for I .

(2)And there exist a valid solution S'' is optimums for I'' of $v'' > v = v^*$- v_n. v''+ $v_n > v +\ v_n = v^*$. It's conflict with S^* is optimums for I .

S^* = (k, $s_1, \dots, s_k$) I = (C , $s_1 , \dots, s_n$) $I' = (C, s_1, \dots, s_{n-1})$ $I'' = ((C - s_n), s_1, \dots, s_{n-1})$
S = (k-1, $s_1, \dots, s_{k-1}$) v= v^* - v_n



Formula

- What is added from S to S*

$$S^* = \begin{cases} S^* \\ S + s_n \end{cases}$$

- How to get from v to v^*

$$v^* = \begin{cases} v^* \\ v^* + v_n \end{cases}$$

- What changes between I and I', I''

  I and I' different in one element s_n with same capacity

  I and I' different in one element s_n with different capacity

- Dynamic Programming Formula

$$\text{KS}(C, n) = \begin{cases} KS(C, n-1) \\ KS(C - v_n, n-1) + v_n \end{cases} \text{ where } \text{KS}(C,0) = 0$$

S^* = (k, $s_1, \ldots, s_k$) S = (k-1, $s_1, \ldots, s_{k-1}$) v= v^* - v_n

Complexity

$$O(state * space * cost\ of\ function)$$
$$= O\big((C) * (n) * (1)\big) = O\ (C * n)$$

Source

- Advanced Algorithms: Dynamic Programming, Spring 2020, Erik Saule
- Mathematical Foundations of Computing: Relations and Functions, Fall 2015, Keith Schwarz

Please cite if you use any part of our solution in your work. This includes, but is not limited to, adaptations, transformations, etc., …

Thank you