Submodular Maximization with a Knapsack Constraint

Alina Ene
Boston University

Joint work with
Huy Nguyen (Northeastern University)
Input:

Items $V = \{e_1, e_2, \ldots, e_n\}$ with costs $c_{e_i}$

Goal:

$$\max_{S \subseteq V} f(S) \quad \text{s.t.} \quad c(S) \leq 1$$

NP-hard, admits a tight $1 - \frac{1}{e}$ approx
Approach 1: Guess most valuable items + Density Greedy
Approach 1: Guess most valuable items + Density Greedy

Order the optimal solution OPT

\[ o_1, o_2, o_3, \ldots, o_k \]

\[ o_i = \arg\max_{o \in \text{OPT}\setminus\{o_1, \ldots, o_{i-1}\}} f(\{o_1, \ldots, o_{i-1}\} \cup \{o\}) - f(\{o_1, \ldots, o_{i-1}\}) \]
Approach 1: Guess most valuable items + Density Greedy

Order the optimal solution $\text{OPT}$

$$o_1, o_2, o_3, \ldots, o_k$$

$$o_i = \arg\max_{o \in \text{OPT}\backslash\{o_1, \ldots, o_{i-1}\}} f(\{o_1, \ldots, o_{i-1}\} \cup \{o\}) - f(\{o_1, \ldots, o_{i-1}\})$$

[Sviridenko '04]

Guess the first 3 items $o_1, o_2, o_3$

Fill up remaining budget using Density Greedy
Approach 1: Guess most valuable items + Density Greedy

Order the optimal solution OPT

\[ o_1, o_2, o_3, \ldots, o_k \]

\[ o_i = \arg\max_{o \in \text{OPT} \setminus \{o_1, \ldots, o_{i-1}\}} f(\{o_1, \ldots, o_i\} \cup \{o\}) - f(\{o_1, \ldots, o_{i-1}\}) \]

[Sviridenko ’04]

Guess the first 3 items \( o_1, o_2, o_3 \)

Fill up remaining budget using Density Greedy

\[ 1 - \frac{1}{e} \] approx in time \( O(n^5) \) (time = function evals)

\[ 1 - \frac{1}{e} - \epsilon \] approx in time \( \tilde{O}(n^4) \) via lazy Density Greedy
Approach 2: Guess (approximate) marginal values

\[ \text{OPT}_1 \]
\[ o_1, o_2, \ldots, o_t \]
large value items

\[ \text{OPT}_2 \]
\[ o_{t+1}, o_{t+2}, \ldots, o_k \]
small value items

Guess (approximately)

The marginal value of every item in \( \text{OPT}_1 \)

The total marginal value of \( \text{OPT}_2 \)

Greedily pick items to meet these marginal values
Approach 2: Guess (approximate) marginal values

[Badanidiyuru, Vondrak ’14]

\[
\max \quad F(x) \\
\text{s.t. } \langle c, x \rangle \leq 1 \\
x \in [0, 1]^n
\]

Large Items: Pack using Continuous Greedy
Small Items: Pack using Continuous Density Greedy

Round the fractional solution without any loss

\[
1 - \frac{1}{e} - \epsilon \quad \text{approx in time } O\left(n^2 \cdot \left(\frac{\log n}{\epsilon}\right)^{\frac{1}{e^8}}\right)
\]
[Badanidiyuru, Vondrak ’14]

Solve the multilinear relaxation in time $\tilde{O}_\epsilon(n^2)$
Round solution without loss in time $\tilde{O}(n)$

Where does the quadratic running time come from?

The algorithm is actually very efficient
It evaluates $F(x)$ only $\tilde{O}_\epsilon(n)$ times
Each $F(x)$ evaluation requires evaluating $fO(n)$ times
Technical challenge

Develop submodular maximization algorithms that use fewer than $n^2$ evaluation queries

Conceptual challenge

Avoid generic uses of the multilinear extension
[Badanidiyuru, Vondrak ’14]

Solve the multilinear relaxation in time $\tilde{O}_\epsilon(n^2)$

Round solution without loss in time $\tilde{O}(n)$

Overall running time: $O\left(n^2 \left(\frac{\log n}{\epsilon}\right)^{\frac{1}{\epsilon^8}}\right)$

Our contributions:

Algorithm for knapsack with $\tilde{O}_\epsilon(n)$ evaluations

Ensure $x$ only has $O_\epsilon(1)$ fractional entries

Each $F(x)$ evaluation takes $O_\epsilon(1)$ value queries

Overall running time: $\left(\frac{1}{\epsilon}\right)^{O\left(\frac{1}{\epsilon^4}\right)} n \log^2 n$
For $p = 1, 2, \ldots, 1/\epsilon$

$$t = \frac{1}{\epsilon^3} \text{ large value items}$$

$$o_1, o_2, \ldots, o_t$$

$$o_{t+1}, o_{t+2}, \ldots, o_k$$

small value items
For $p = 1, 2, \ldots, 1/\epsilon$

For $i = 1, 2, \ldots, t$

Guess $v_{p,i} \approx F(x \lor 1_{o_i}) - F(x)$

Select $e_{p,i}$ with marginal value $\geq v_{p,i}$ and min cost

Update $x \leftarrow x + \epsilon 1_{e_{p,i}}$
For $p = 1, 2, \ldots, 1/\epsilon$

For $i = 1, 2, \ldots, t$

Guess $v_{p,i} \approx F(x \lor 1_{o_i}) - F(x)$

Select $e_{p,i}$ with marginal value $\geq v_{p,i}$ and min cost

Update $x \leftarrow x + \epsilon 1_{e_{p,i}}$

Guess $w_p \approx F(x \lor 1_{OPT_2}) - F(x)$

Use Density Greedy to integrally select a set $S_p$

with total gain $F(x \lor 1_{S_p}) - F(x) \approx \epsilon w_p$

Update $x \leftarrow x \lor 1_{S_p}$
The algorithm constructs a solution $x$ with:

$$F(x) \geq \left(1 - \frac{1}{e} - \epsilon\right) f(OPT)$$
The algorithm constructs a solution $x$ with:

$$F(x) \geq \left(1 - \frac{1}{\epsilon} - \epsilon\right) f(\text{OPT})$$

$$\frac{1}{\epsilon^4}$$

fractional entries that can be charged to $\text{OPT}_1$

For $p = 1, 2, \ldots, 1/\epsilon$

For $i = 1, 2, \ldots, t$ ($t = 1/\epsilon^3$)

Guess $v_{p,i} \approx F(x \lor 1_{o_i}) - F(x)$

Select $e_{p,i}$ with marginal value $\geq v_{p,i}$ and min cost

Update $x \leftarrow x + \epsilon 1_{e_{p,i}}$
Round: preserve value, cost $\leq c(OPT_1)$
Round: preserve value, cost $\leq c(\text{OPT}_1)$

Intuition: similar to a partition matroid
Round: preserve value, cost $\leq c(OPT_1)$

Intuition: similar to a partition matroid

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(1_{e_1} - 1_{e_2}) \text{ where } \mathbb{E}x[\delta] = 0$$
Round: preserve value, cost \( \leq c(\text{OPT}_1) \)

Intuition: similar to a partition matroid

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

\[ x' \leftarrow x + \delta(1_{e_1} - 1_{e_2}) \text{ where } \mathbb{E}x[\delta] = 0 \]
Round: preserve value, cost $\leq c(\text{OPT}_1)$

Intuition: similar to a partition matroid

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

$x' \leftarrow x + \delta(1_{e_1} - 1_{e_2})$ where $\mathbb{E}x[\delta] = 0$
Round: preserve value, cost \( \leq c(OPT_1) \)

Intuition: similar to a partition matroid

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

\[
x' \leftarrow x + \delta (1_{e_1} - 1_{e_2}) \quad \text{where } \mathbb{E}x[\delta] = 0
\]
Round: preserve value, cost $\leq c(\text{OPT}_1)$

Intuition: similar to a partition matroid

decreasing costs

Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(1_{e_1} - 1_{e_2})$$

where $\mathbb{E}x[\delta] = 0$
Round: preserve value, cost $\leq c(OPT_1)$

Intuition: similar to a partition matroid

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

$x' \leftarrow x + \delta(1_{e_1} - 1_{e_2})$ where $\mathbb{E}x[\delta] = 0$
Round: preserve value, cost $\leq c(OPT_1)$

Intuition: similar to a partition matroid

Sort the costs of fractional items in decreasing order

Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(1_{e_1} - 1_{e_2})$$ where $\mathbb{E}x[\delta] = 0$
Round: preserve value, cost $\leq c(OPT_1)$

Intuition: similar to a partition matroid

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

$x' \leftarrow x + \delta(1_{e_1} - 1_{e_2})$ where $\text{Ex}[\delta] = 0$
Round: preserve value, cost $\leq c(OPT_1)$

Intuition: similar to a partition matroid

decreasing costs

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(1_{e_1} - 1_{e_2}) \text{ where } \mathbb{E}x[\delta] = 0$$
Round: preserve value, cost $\leq c(\text{OPT}_1)$

Intuition: similar to a partition matroid

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(1_{e_1} - 1_{e_2}) \text{ where } \mathbb{E}x[\delta] = 0$$
Round: preserve value, cost $\leq c(\text{OPT}_1)$

Intuition: similar to a partition matroid

decreasing costs

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta(1_{e_1} - 1_{e_2})$$ where $\mathbb{E}x[\delta] = 0$
Round: preserve value, cost $\leq c(\text{OPT}_1)$

Intuition: similar to a partition matroid

decreasing costs

Sort the costs of fractional items in decreasing order
Move fractional mass b/w the 2 items with highest cost

$$x' \leftarrow x + \delta (1_{e_1} - 1_{e_2}) \text{ where } \mathbb{E}x[\delta] = 0$$
For $p = 1, 2, \ldots, 1/\epsilon$

For $i = 1, 2, \ldots, t$

Guess $v_{p,i} \approx F(x \lor 1_{o_i}) - F(x)$

Select $e_{p,i}$ with marginal value $\geq v_{p,i}$ and min cost

Update $x \leftarrow x + \epsilon 1_{e_{p,i}}$

Guess $w_p \approx F(x \lor 1_{\text{OPT}_2}) - F(x)$

Use Density Greedy to integrally select a set $S_p$

with total gain $F(x \lor 1_{S_p}) - F(x) \approx \epsilon w_p$

Update $x \leftarrow x \lor 1_{S_p}$
For $p = 1, 2, \ldots, 1/\epsilon$

Need to ensure that Density Greedy picks items with cost at most $1 - c(\text{OPT}_1)$

Guess $w_p \approx F(x \lor 1_{\text{OPT}_2}) - F(x)$

Use Density Greedy to integrally select a set $S_p$ with total gain $F(x \lor 1_{S_p}) - F(x) \approx \epsilon w_p$

Update $x \leftarrow x \lor 1_{S_p}$
\[ t = \frac{1}{\epsilon^3} \text{ large value items} \]

Can assume that each item in $\text{OPT}_2$ has small cost

\[ c_o \leq \epsilon^2 (1 - c(\text{OPT}_1)) \quad \forall o \in \text{OPT}_2 \]
Can assume that each item in $\text{OPT}_2$ has small cost

$$c_o \leq \epsilon^2 (1 - c(\text{OPT}_1)) \quad \forall o \in \text{OPT}_2$$

Each item $o \in \text{OPT}_2$ satisfies

$$f(\text{OPT}_1 \cup \{o\}) - f(\text{OPT}_1) \leq \epsilon^3 f(\text{OPT})$$

There are at most $1/\epsilon^2$ items $o \in \text{OPT}_2$ with cost

$$c_o > \epsilon^2 (1 - c(\text{OPT}_1))$$

Discard all of them and lose only $\epsilon f(\text{OPT})$
Can assume that each item in $\text{OPT}_2$ has small cost

$$c_o \leq \epsilon^2 (1 - c(\text{OPT}_1)) \quad \forall o \in \text{OPT}_2$$
\[ t = \frac{1}{\epsilon^3} \] large value items

Can assume that each item in \( \text{OPT}_2 \) has small cost

\[ c_o \leq \epsilon^2 (1 - c(\text{OPT}_1)) \quad \forall o \in \text{OPT}_2 \]

If each item in \( \text{OPT}_2 \) also has small marginal gain

\[ F(x \vee 1_o) - F(x) \] then Density Greedy items will fit
Can assume that each item in $\text{OPT}_2$ has small cost
\[ c_o \leq \epsilon^2 (1 - c(\text{OPT}_1)) \quad \forall o \in \text{OPT}_2 \]

If each item in $\text{OPT}_2$ also has small marginal gain
$F(x \lor 1_o) - F(x)$ then Density Greedy items will fit

Items in $\text{OPT}_2$ only have small gain on top of $\text{OPT}_1$
Marginal gain on top of $x$ could be large

\[ t = \frac{1}{\epsilon^3} \quad \text{large value items} \]

\[ o_{t+1}, o_{t+2}, \ldots, o_k \quad \text{small value items} \]
Can assume that each item in $\text{OPT}_2$ has small cost

$$c_o \leq \epsilon^2 (1 - c(\text{OPT}_1)) \quad \forall o \in \text{OPT}_2$$

If each item in $\text{OPT}_2$ also has small marginal gain

$$F(x \lor 1_o) - F(x)$$

then Density Greedy items will fit

Items in $\text{OPT}_2$ only have small gain on top of $\text{OPT}_1$

Marginal gain on top of $x$ could be large

Can fix this issue using more guessing, similarly to $\text{OPT}_1$
Summary and Open Questions

Overall running time: \( \left( \frac{1}{\varepsilon} \right)^{O\left(\frac{1}{\varepsilon^4}\right)} n \log^2 n \)

Improve the dependency on \( \varepsilon \)? (FPTAS for linear fns)

Faster running times for matroid constraints?

[BV ’14] \( O\left(\frac{rn}{\varepsilon^4} \log^2 \left(\frac{r}{\varepsilon}\right)\right) \) time

Partition matroid is a natural (and challenging) case