# Submodular Optimization

# Contents

# Introduction

This lecture is about:

- discrete optimization
- submodular function/optimization
  (has performance guarantee)

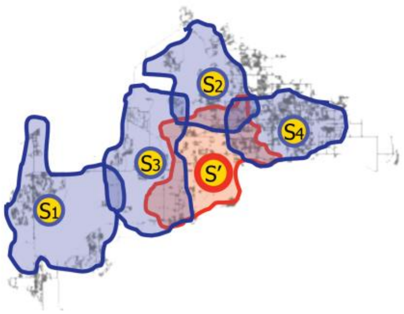
# Example – sensor placement

Problem setting:

- Deploy sensors in the water distribution network to detect contamination
- Different sensors have different coverage abilities

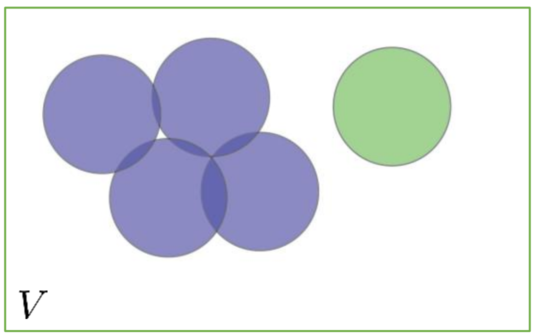
Question:

- Which sensor(s) should be selected if there is upper limit of # of selected sensors?

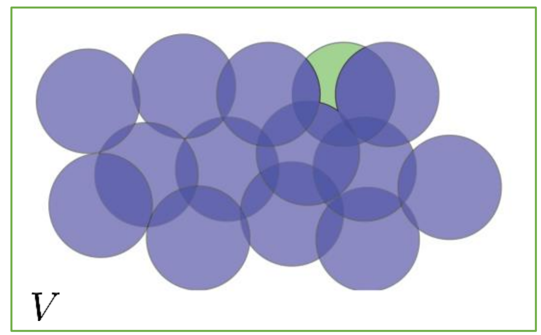


# Example – set cover

$f(S) = \bigcup_{e \in S} \text{area}(\{e\})$




$A$ (purple) $e$ (green)




$B$ (purple) $e$ (green)

$V$: ground set
$A \subseteq B$
$e \in V \setminus B$

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$$

marginal gain decrease

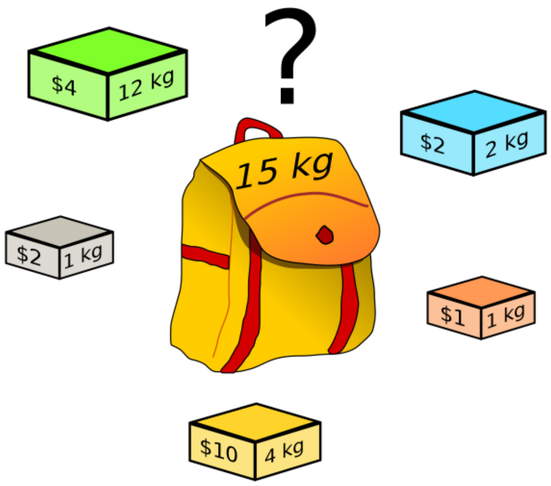
# Example - knapsack problem

Problem setting:

- Given a set of items, each with a weight and a value,

Problem:

- Which items should be included to maximize the value such that the total weight is $<=$ a limit




*Note: this is a modular problem, but still can be fit in our problem formulation which will be introduced later.*

Is there a general problem formulation that covers this type of problem?

Yes. Submodular optimization

# Definition – submodular optimization

Maximize or minimize a submodular function with(or without) constraints considered.

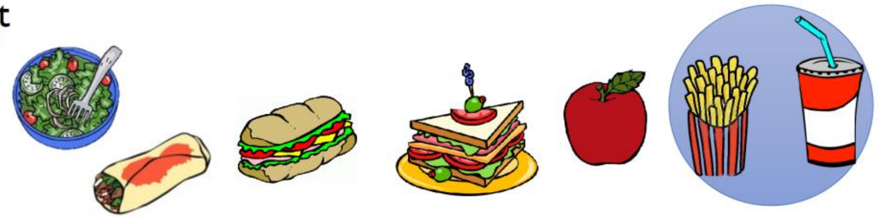
$\boxed{\text{Set function:}}$ $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$, where $\mathcal{V}$ is ground set.

$\boxed{\text{Submodular function:}}$ $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$, if $A \subseteq B$ and $e \in V \setminus B$

marginal gain/cost decrease

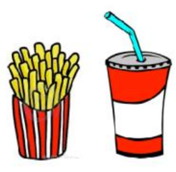
# Set function – example 1

ground set

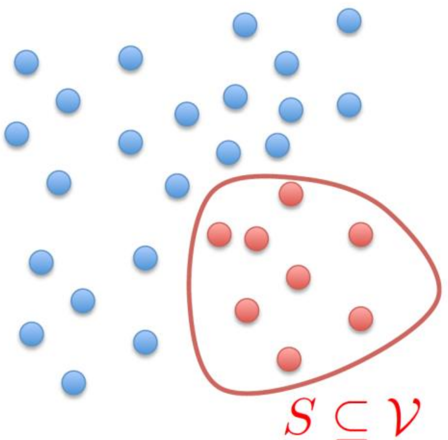
$\mathcal{V} = \{$  $\}$

objective function $\quad F : 2^{\mathcal{V}} \to \mathbb{R}$

---

$F($  $) =$ cost of buying items together, or utility, or probability

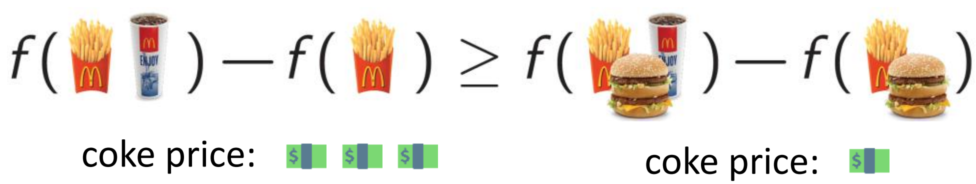
# Set function – example 2




- ground set $\mathcal{V}$
- (scoring) function $F : 2^{\mathcal{V}} \to \mathbb{R}_+$

# Submodular function – example 1

Consumer costs are very often submodular.

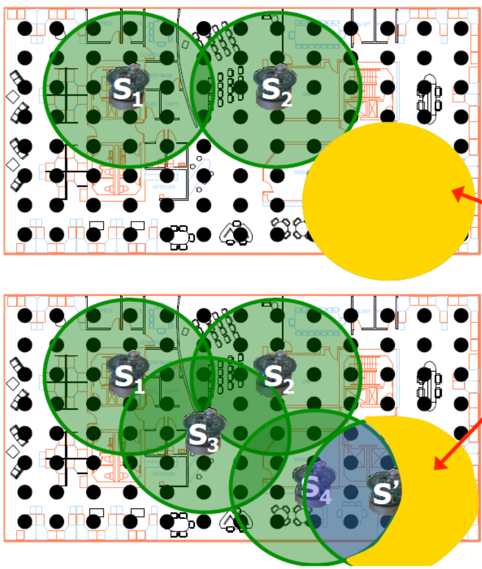



where $f(\cdot)$ is a cost function

$\boxed{\text{marginal cost decrease}}$

Note: The additional cost of a coke is, say, free if you add it to fries and a hamburger, but when added just to an order of fries, the coke is not free.

# Submodular function – example 2




A={$s_1$,$s_2$}

F(A U {s'}) – F(A)

≥

F(B U {s' }) – F(B)

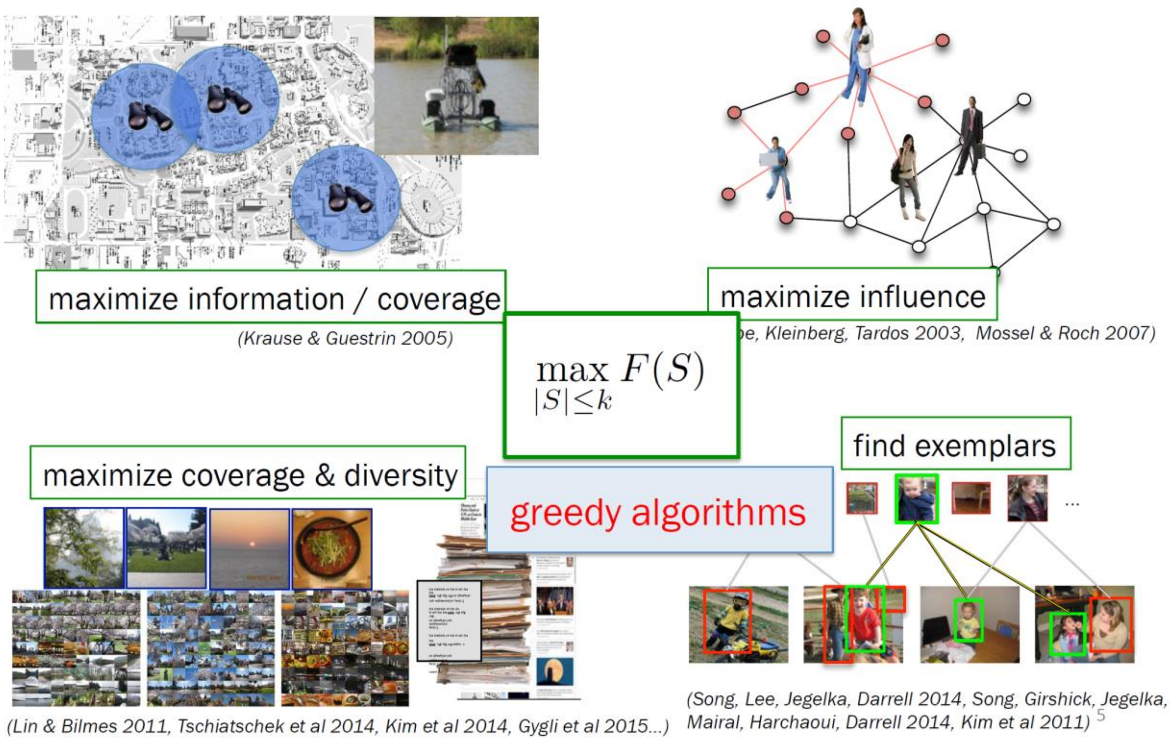
B = {$s_1$,$s_2$,$s_3$,$s_4$}

objective function $F : 2^{\mathcal{V}} \rightarrow \mathbb{R}$

the total covered area.

$\boxed{\text{marginal gain decrease}}$

# Submodular optimization – examples



# Submodular definition

A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A, B \subseteq V$,

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$

or

A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A \subseteq B \subset V$, and $v \in V \setminus B$, we have that:

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$$

diminishing returns property

# Good to know - supermodular

A function $f : 2^V \to \mathbb{R}$ is supermodular if for any $A, B \subseteq V$, that:

$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$$

or

A function $f : 2^V \to \mathbb{R}$ is supermodular if for any $A \subseteq B \subset V$, and $v \in V \setminus B$, we have that:

$$f(A \cup \{v\}) - f(A) \leq f(B \cup \{v\}) - f(B)$$

# Submodular vs. supermodular

- Submodular and supermodular functions are closely related.
- In fact, $f$ is submodular iff $-f$ is supermodular.

# More definitions

submodular: $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$

modular: $f(A) + f(B) = f(A \cap B) + f(A \cup B)$

modular can be viewed as special of submodular

subadditive: $f(A) + f(B) \geq f(A \cup B)$

This means that the whole is less than the sum of the parts.

supadditive: $f(A) + f(B) \leq f(A \cup B)$

This means that the whole is greater than the sum of the parts.

# submodular maximization - greedy

Problem: $\max f(X)$ subject to $|X| \leq k$

Algorithm:

Set $S_0 \leftarrow \emptyset$ ;
**for** $i \leftarrow 0 \ldots |E| - 1$ **do**
  Choose $v_i$ as follows:
  $v_i \in \left\{ \operatorname{argmax}_{v \in V \setminus S_i} f(\{v\}|S_i) \right\} = \left\{ \operatorname{argmax}_{v \in V \setminus S_i} f(S_i \cup \{v\}) \right\}$ ;
  Set $S_{i+1} \leftarrow S_i \cup \{v_i\}$ ;

performance:

$$f(S_\ell) \geq (1 - e^{-\ell/k}) \max_{S:|S| \leq k} f(S)$$

# Distributed resilient action selection

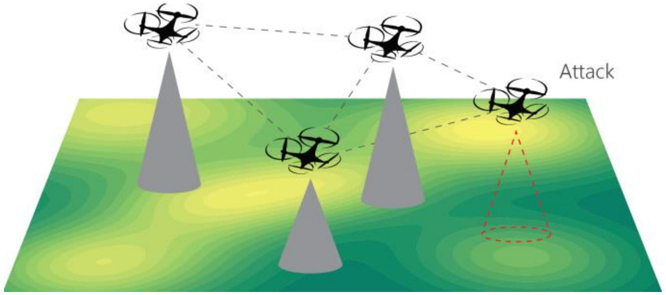
Application: Environmental exploration

Settings:

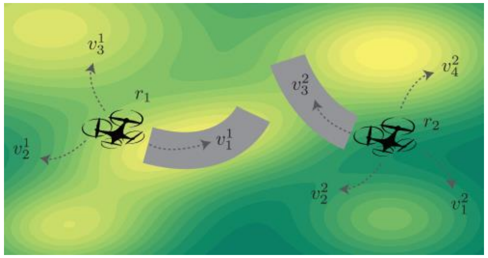
- Environment: importance map
- Robots: have several candidate actions, can only choose one.
- Task: collect importance
- Attacker: attack robots
- Distributed communication

Assumption: # of attackers is limited

Goal: protect system against worst-case attack




[RA-L'21]

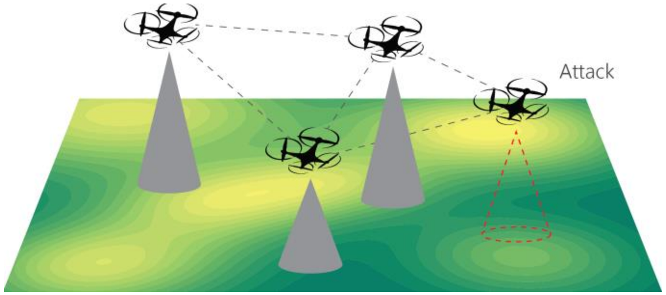



[RA-L'21]

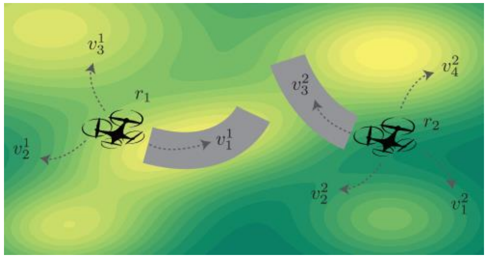
# Distributed resilient action selection

Questions:

- how to counter attacks without knowing attackers' behavior?
- work for distributed scenario?
- has performance guarantees?
- etc.




[RA-L'21]




[RA-L'21]

# Distributed resilient action selection

Robots' selected action set

Explored importance

Attackers' action set

$$
\begin{aligned}
\underset{\mathcal{S}\subseteq\mathcal{V}}{\text{maximize}} \quad & \min_{\mathcal{F}\subseteq\mathcal{S}} f(\mathcal{S}\setminus\mathcal{F}) \\
\text{subject to} \quad & |\mathcal{S}\cap\mathcal{V}_i| = 1, \forall\, i \in \mathcal{R}, \\
& |\mathcal{F}| \leq K,
\end{aligned}
$$

Each robot selects one action

Attackers' action set

Max # of attackers

- "min": worst-case attack
- Robots: want to maximize reward
- Attackers: want to minimize reward

# Distributed resilient action selection algorithm

Phase I: greedy without considering others $\mathcal{S}_1$

- Each robot selects the action greedily. $\arg\max_{v \in \mathcal{V}_i} f(v)$
- Communicate with neighbors to achieve consensus on *K* actions

Phase II: greedy with considering others $\mathcal{S}_2$

- The rest of (*N*-*K*) robots are active
- Active robots greedily selects action from its action set $\underset{v \in \mathcal{V}_i}{\arg\max} f(v \cup \mathcal{S}_1 \cup \mathcal{S}_2)$
- Communicate with neighbors: resolve conflict

Final solution: $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$

# Distributed resilient action selection performance

$\boxed{\textit{Performance:}}$ The approximation ratio is

$$\boxed{f(\mathcal{S} \setminus \mathcal{F}^\star)} \geq \max\{\frac{1-c_f}{1+c_f}, \frac{1}{1+K}, \frac{1}{|\mathcal{R}|-K}\} \boxed{f(\mathcal{S}^\star \setminus \mathcal{F}^\star)}.$$

solution

Optimal

Curvature: measure how far *f* is from being additive.

$$c_f = 1 - \min_{v \in \mathcal{V}} \frac{f(\mathcal{V}) - f(\mathcal{V} \setminus \{v\})}{f(v)}$$

# Current status of submodular maximization

- Generally, NP hard.

- For monotone submodular function,
  - In 1978, it was shown that greedy algorithm guarantees (1-1/e) approximation for cardinality constraint.
  - In 2007, (1-1/e) approximation was proved for matroid constraint.

- For non-negative submodular function,
  - In 2011, 2/5approximation was shown for problems without constraint.

# References?

- Derive the basic properties yourself. Don't rely on tutorial (including my lecture).

- A feasible way…
  - (1) Read Chapters 1, 2, 3 and 10 of "Learning with Submodular Functions: A Convex Optimization Perspective" (233 pages long)
    - You will have good knowledge to understand minimization problems.
  - (2) Read Chapter 11 of the paper mentioned above.
    - You will get some sense about maximization.
  - (3) Read the two papers by Vondrak, "Optimal approximation for the Submodular Welfare Problem in the value oracle model" and "maximizing non-monotone submodular functions".
  - (4) Continue to read papers according to your need…