Tutorial: Evolutionary Submodular Optimisation



Presenters:
Aneta Neumann[1], Frank Neumann[1], Chao Qian[2]

[1] Optimisation and Logistics, The University of Adelaide, Australia.
[2] School of Artificial Intelligence, Nanjing University, China.

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07...$15.00
https://doi.org/10.1145/3449726.3461433

Our task:

Given a function $f : X \rightarrow R$
and $D \subseteq X$ "set of feasible solutions"

Find $\arg\max_{x \in D} f(x)$

General purpose algorithms that can be applied without problem knowledge

Why General Purpose Algorithms?

- Algorithms are the heart of every nontrivial computer application.
- For many problems we know good or optimal algorithms
  - Sorting
  - Shortest paths
  - Minimum spanning trees
- What about a new or complex problems?
- Often there are no good problem specific algorithms.

Points that may rule out problem specific algorithms

- Problems that are rarely understood.
- Quality of solutions is determined by simulations.
- Problem falls into the black box scenario.

- Not enough resources such as time, money, knowledge.

General purpose algorithms are often a good choice.

General purpose algorithms for
optimizing a function $f: X \rightarrow R$

1. Choose a representation for the elements in X.
2. Fix a function to evaluate the quality.
   (might be different from f)
3. Define operators that produce new elements.

Evolutionary algorithms (EAs)

- Evolutionary algorithms are general purpose algorithms.
- follow Darwin's principle (survival of the fittest).
- work with a set of solutions called population.
- parent population produces offspring population by variation operators (mutation, crossover).
- select individuals from the parents and children to create new parent population.
- Iterate the process until a "good solution" has been found.
- EAs are adaptive and often yield good solutions for complex, dynamic and/or stochastic problems

Motivation

- Want to understand a wide class of problems that evolutionary algorithms can solve or approximate well.
- Consider submodular functions which allow to model a wide range of important optimisation problems.
- Submodular functions can be considered as the discrete counterpart of convexity in the continuous domain (Lovasz, 1983).

Submodular Functions

- Let $X=\{x_1, \ldots., x_n\}$ be a ground set
- *Submodular functions:* A function f is submodular iff $f(A \cup B)+f(A \cap B) \leq f(A)+f(B)$ for all $A, B \subseteq X$.
- Alternative definition of submodularity:

$$A \subseteq B \subseteq X \text{ and } x \in X \setminus B,\ f(B \cup \{x\})-f(B) \leq f(A \cup \{x\})-f(A).$$

Maximizing submodular functions is NP-hard and also NP-hard to approximate.

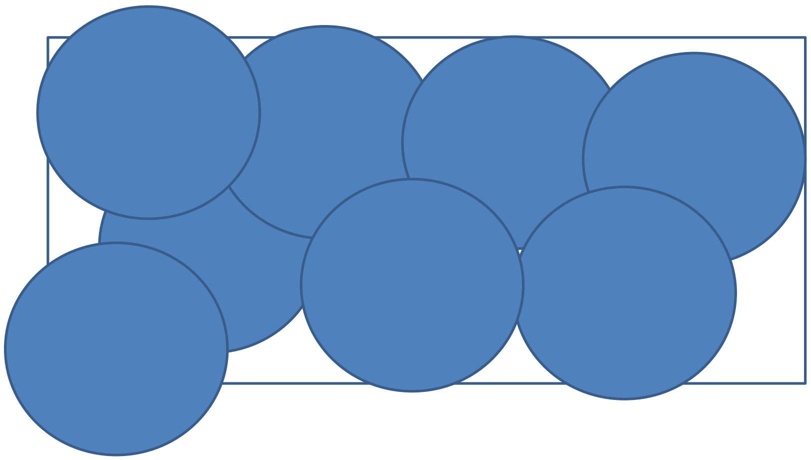
Important subclasses:

- *Monotone functions:* A function is monotone iff $f(A) \leq f(B)$ for all $A \subseteq B$.

We call f symmetric iff $f(A)=f(X \setminus A)$ for all $A \subseteq X$.

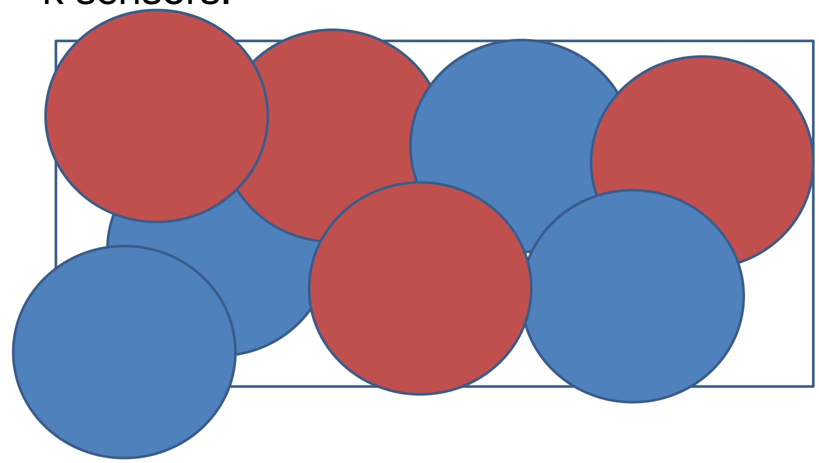
Example: Sensor placement

Cover the largest possible area by selecting k sensors:



Example: Sensor placement

Cover the largest possible area by selecting k sensors:



Example Max Cut

- Given an undirected graph G=(V, E), find a partitioning of the vertices such that the number of edges crossing the two partitions is maximal.

- A is a set of nodes chosen for the first partition. Function f(A) counts the number of edges between A and V \ A.
- f is symmetric, submodular, but not monotone.

Matroids

A matroid is a pair (X, I) composed of a ground set X and a non-empty collection I of subsets of X satisfying (1) If $A \in I$ and $B \subseteq A$ then $B \in I$ and (2) If $A, B \in I$ and $|A| > |B|$ then $B + x \in I$ for some $x \in A \setminus B$. The sets in I are called *independent*, the *rank* of a matroid is the size of any maximal independent set.

Example:

- For given graph G=(V,E), M=(E, F) where F is the set of all forests (subset of edges not containing cycles) is a matroid. Maximal independent sets are spanning trees (rank n-1).
- Given X all subsets of cardinality at most k build the uniform matroid.

Some Examples of Submodular Functions

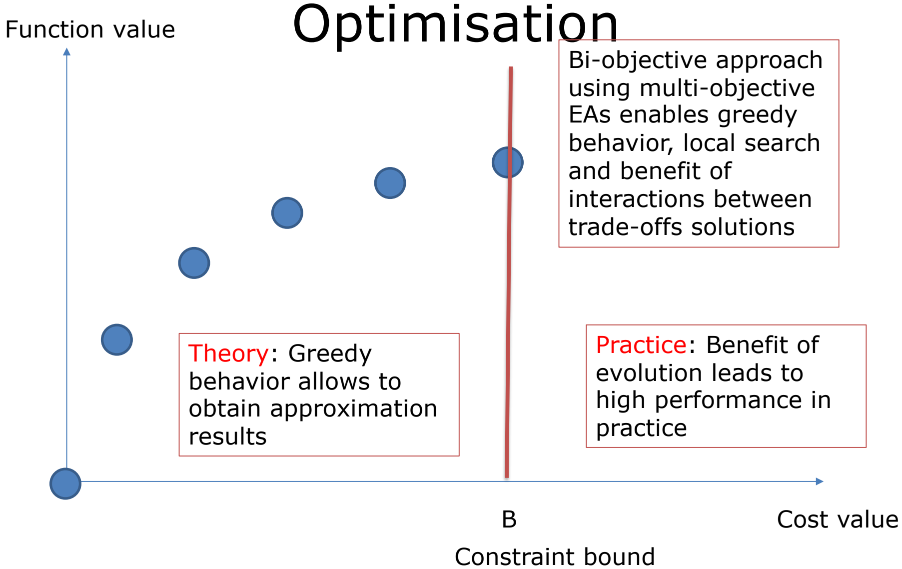
- *Linear functions:* All linear functions $f\colon 2^X \to \mathbb{R}$ with $f(A) = \sum_{i \in A} w_i$ for some weights $w\colon X \to \mathbb{R}$ are submodular. If $w_i \geq 0$ for all $i \in X$, then f is also monotone.
- *Cut:* Given a graph $G = (V, E)$ with nonnegative edge weights $w\colon E \to \mathbb{R}_{\geq 0}$. Let $\delta(S)$ be the set of all edges that contain both a vertex in S and $V \setminus S$. The cut function $w(\delta(S))$ is symmetric and submodular but not monotone.
- *Coverage:* Let the ground set be $X = \{1, 2, \ldots, n\}$. Given a universe U with n subsets $A_i \subseteq U$ for $i \in X$, and a non-negative weight function $w\colon U \to \mathbb{R}_{\geq 0}$. The coverage function $f\colon 2^X \to \mathbb{R}$ with $f(S) = |\bigcup_{i \in S} A_i|$ and the weighted coverage function f' with $f'(S) = w(\bigcup_{i \in S} A_i) = \sum_{u \in \bigcup_{i \in S} A_i} w(u)$ are monotone submodular.
- *Rank of a matroid:* The rank function $r(A) = \max\{|S|\colon S \subseteq A, S \in \mathcal{I}\}$ of a matroid $(X, \mathcal{I})$ is monotone submodular.

Submodular Optimisation

Research in this area can be characterized by the type of

- Functions to be optimized
  - Submodular / close to submodular
  - monotone / non-monotone
  - Additional function characteristics
- Types of constraints
  - Uniform, linear constraints
  - General cost constraints
  - Matroid / partition constraints.
  - Other types of constraints

Bi-objective approach / Pareto Optimisation



GSEMO

- Given submodular function f, solutions are encoded as bitstrings of length n.

Algorithm 1: GSEMO Algorithm

choose $x \in \{0,1\}^n$ uniformly at random
determine $g(x)$
$P \leftarrow \{x\}$
repeat
  choose $x \in P$ uniformly at random
  create x' by flipping each bit x_i of x with probability $1/n$
  determine $g(x')$
  if x' is not strictly dominated by any other search point in P **then**
    include x' into P
    delete all other solutions $z \in P$ with $g(z) \leq g(x')$ from P
until *stop*

- Maximize bi-objective function $g(x)=(z(x), |x|_0)$, where $z(x)=f(x)$ iff x is feasible and $z(x)=-1$ otherwise
- **Analyze expected time (number of fitness evaluations) to obtain good approximations**

Monotone submodular functions under uniform constraint

A solution x is feasible iff its has at most k elements (1-bits), i.e.

$$F = \{x \mid x \in X \wedge |x|_1 \leq k\}$$

is the set of feasible solutions.

Result (Friedrich, Neumann (ECJ 2015)):

GSEMO achieves a (1-1/e)-approximation in expected time $O(n^2(k + \log n))$.

Proof Idea

- GSEMO obtains empty set in expected time $O(n^2 \log n)$.
- Afterwards mimics greedy approach (Nemhauser et al 1978) and obtains for each j, $0 \leq j \leq k$, a solution X_j with

$$f(X_j) \geq \left(1 - \left(1 - \frac{1}{k}\right)^j\right) \cdot f(\mathrm{OPT}),$$

where f(OPT) is value of feasible optimal solution.

Key induction argument:

– Assume that we already have $f(X_j) \geq \left(1 - (1 - \frac{1}{k})^j\right) \cdot f(OPT), 0 \leq j \leq i.$

Let δ_{i+1} be the increase in f that we obtain when choosing the solution $x \in P$ with $|x|_1 = i$ for mutation and inserting the element corresponding to the largest possible increase.

Due to monotonicity and submodularity, we have $f(OPT) \leq f(X_i \cup OPT) \leq f(X_i) + k\delta_{i+1}$ which implies $\delta_{i+1} \geq \frac{1}{k} \cdot (f(OPT) - f(X_i))$.

Gives X_{i+1} with $f(X_{i+1}) \geq f(X_i) + \frac{1}{k}(f(OPT) - f(X_i)) \geq \left(1 - (1 - \frac{1}{k})^{i+1}\right) \cdot f(OPT)$.

- X_k is (1-1/e)-approximation and obtained after $O(n^2 k)$ steps.

Non-monotone symmetric under Matroid Constraints

Given k matroids $M_1, \ldots, M_k$ together with their independent systems $I_1, \ldots, I_k$, we consider the problem

$$\max\left\{f(x) \mid x \in F := \bigcap_{j=1}^k I_j\right\}.$$

We assume that f is symmetric, submodular and non-negative, but not necessarily monotone.

- For this setting, a local search capability is beneficial to obtain good approximations.
- In particular, dependent on the number of matroids, a local improvement in a neighborhood dependent on k can be obtained if the current solution is not of sufficient quality (Lee et al, STOC 2009).

Non-monotone symmetric under Matroid Constraints

Result (Friedrich, Neumann (ECJ 2015)):
GSEMO achieves a $\left(\frac{1}{(k+2)(1+\epsilon)}\right)$-approximation in expected time $O\left(\frac{1}{\epsilon} n^{k+6} \log n\right)$.
Proof idea:

- In expected time $O(n^2 \log n)$, GSEMO produces the search point 0^n.
- Introducing the element with the largest gain gives a solution of quality at least OPT/n.
- Afterwards from the currently best feasible solution x, in expected time $O(n^{k+2})$ a solution y with $f(y) \geq (1 + \varepsilon/n^4) \cdot f(x)$ can be produced if stated approximation guarantee has not yet been obtained.
- Total number of such local improvements required to obtain approximation is at most

$$\log_{(1+\frac{\epsilon}{n^4})} \frac{OPT}{OPT/n} = O\left(\frac{1}{\epsilon} n^4 \log n\right).$$

Remark: k=1 gives $(1/(3(1+\varepsilon)))$-approximation for Max-Cut

Approximately Submodular Functions

Approximately submodular application

Sparse regression [Tropp, TIT'04]: given observation variables $V = \{v_1, \dots, v_n\}$, a predictor variable z and a budget B, to find a subset $X \subseteq V$ such that

$$max_{X \subseteq V} \quad R^2_{z,X} = \frac{\mathrm{Var}(z) - \mathrm{MSE}_{z,X}}{\mathrm{Var}(z)} \quad s.t. \quad |X| \le B$$

$\mathrm{Var}(z)$: variance of z

$\mathrm{MSE}_{z,X}$: mean squared error of predicting z by using observation variables in X

$R^2_{z,X}$: squared multiple correlation, which is approximately submodular

observation variables

predictor variable z

	Corr.	Dis.	LR	···	···	AIC.	BIC	RF.
x1	0.28	0.46	1	···	···	0.22	0.63	1
x2	0.31	0.59	0.64	···	···	0.58	0.56	1
x3	0.11	0.02	0.53	···	···	0.43	0.01	1
x4	0.1	0.1	0.64	···	···	0.73	0.92	1
x5	0.02	0.15	0.33	···	···	0.56	0.36	0.78
x6	0.36	0.02	0.01	···	···	0.32	0.02	0.22
x7	0.2	0.2	0.21	···	···	0.21	0.02	0.11
x8	0.1	0.03	0.32	···	···	0.33	0.51	0.44
x9	0.32	0.1	0.2	···	···	0.06	0.66	0
x10	0.24	0	0.02	···	···	0.6	0.03	0.33
x11	0.12	0.45	0.44	···	···	0.64	0.45	1
x12	0.36	0.58	0.12	···	···	0.73	0.58	0.67
x13	0.2	0.02	0.24	···	···	0.34	0.02	0.89
x14	0.24	0.92	0.33	···	···	0.24	0.93	0.56

a subset X of observation variables

	Corr.	Dis.	LR	···	···	AIC.	BIC	RF.
x1	0.28	0.46	1	···	···	0.22	0.63	1
x2	0.31	0.59	0.64	···	···	0.58	0.56	1
x3	0.11	0.02	0.53	···	···	0.43	0.01	1
x4	0.1	0.1	0.64	···	···	0.73	0.92	1
x5	0.02	0.15	0.33	···	···	0.56	0.36	0.78
x6	0.36	0.02	0.01	···	···	0.32	0.02	0.22
x7	0.2	0.2	0.21	···	···	0.21	0.02	0.11
x8	0.1	0.03	0.32	···	···	0.33	0.51	0.44
x9	0.32	0.1	0.2	···	···	0.06	0.66	0
x10	0.24	0	0.02	···	···	0.6	0.03	0.33
x11	0.12	0.45	0.44	···	···	0.64	0.45	1
x12	0.36	0.58	0.12	···	···	0.73	0.58	0.67
x13	0.2	0.02	0.24	···	···	0.34	0.02	0.89
x14	0.24	0.92	0.33	···	···	0.24	0.93	0.56

Submodular ratio

Submodular [Nemhauser et al., MP'78] :

$$\forall X \subseteq Y \subseteq V, v \notin Y: \ f(X \cup \{v\}) - f(X) \ge f(Y \cup \{v\}) - f(Y);$$

$$\text{or } \forall X \subseteq Y \subseteq V: \ f(Y) - f(X) \le \textstyle\sum_{v \in Y \setminus X} f(X \cup \{v\}) - f(X)$$

Submodular ratio [Das & Kempe, ICML'11; Zhang & Vorobeychi, AAAI'16] :

$$\alpha_f = \min_{X \subseteq Y, v \notin Y} \frac{f(X \cup \{v\}) - f(X)}{f(Y \cup \{v\}) - f(Y)}$$

$$\gamma_{U,k}(f) = \min_{X \subseteq U, Y: |Y| \le k, X \cap Y = \emptyset} \frac{\sum_{v \in Y} f(X \cup \{v\}) - f(X)}{f(X \cup Y) - f(X)}$$

Characterize to what extent a set function f satisfies the submodular property, i.e., the degree of approximate submodularity

For example, when f is monotone,

- $\alpha_f \in [0,1]$, the larger, more close to submodular
- f is submodular if and only if $\alpha_f = 1$

Submodular ratio

Submodular [Nemhauser et al., MP'78] :

$$\forall X \subseteq Y \subseteq V, v \notin Y: \ f(X \cup \{v\}) - f(X) \ge f(Y \cup \{v\}) - f(Y);$$

$$\text{or } \forall X \subseteq Y \subseteq V: \ f(Y) - f(X) \le \textstyle\sum_{v \in Y \setminus X} f(X \cup \{v\}) - f(X)$$

Submodular ratio [Das & Kempe, ICML'11; Zhang & Vorobeychi, AAAI'16] :

$$\alpha_f = \min_{X \subseteq Y, v \notin Y} \frac{f(X \cup \{v\}) - f(X)}{f(Y \cup \{v\}) - f(Y)}$$

$$\gamma_{U,k}(f) = \min_{X \subseteq U, Y: |Y| \le k, X \cap Y = \emptyset} \frac{\sum_{v \in Y} f(X \cup \{v\}) - f(X)}{f(X \cup Y) - f(X)}$$

Characterize to what extent a set function f satisfies the submodular property, i.e., the degree of approximate submodularity

For example, when f is monotone,

- $\forall U, k: \gamma_{U,k}(f) \in [0,1]$, the larger, more close to submodular
- f is submodular if and only if $\forall U, k: \gamma_{U,k}(f) = 1$

Submodular ratio

Submodular ratio [Das & Kempe, ICML'11; Zhang & Vorobeychi, AAAI'16] : characterize to what extent a general set function satisfies the submodular property

$$\alpha_f = \min_{X \subseteq Y, v \notin Y} \frac{f(X \cup \{v\}) - f(X)}{f(Y \cup \{v\}) - f(Y)}$$

$$\gamma_{U,k}(f) = \min_{X \subseteq U, Y: |Y| \le k, X \cap Y = \emptyset} \frac{\sum_{v \in Y} f(X \cup \{v\}) - f(X)}{f(X \cup Y) - f(X)}$$

Lower bounds on submodular ratio for some concrete applications

- Sparse regression: $\gamma_{U,k}(f) \ge \lambda_{min}(\mathrm{C}, |U| + k)$ [Das & Kempe, ICML'11]
- Sparse support selection: $\gamma_{U,k}(f) \ge m/M$ [Elenberg et al., Annals of Statistics'18]
- Bayesian experimental design [Bian et al., ICML'17]:

$$\gamma_{U,k}(f) \ge \beta^2 / (\|\mathrm{V}\|^2 (\beta^2 + \sigma^{-2} \|\mathrm{V}\|^2))$$

- Determinantal function maximization [Qian et al., IJCAI'18]:

$$\alpha_f \ge (\lambda_n(\mathrm{A}) - 1) / \left((\lambda_1(\mathrm{A}) - 1) \prod_{i=1}^{n-1} \lambda_i(\mathrm{A}) \right)$$

Pareto optimization for approximately submodular f

The POSS algorithm [Qian, Yu and Zhou, NIPS'15]

$$max_{\boldsymbol{x} \in \{0,1\}^n} \; f(\boldsymbol{x}) \quad s.t. \quad |\boldsymbol{x}| \le B \qquad \text{original}$$

Transformation: $\Downarrow$

$$min_{\boldsymbol{x} \in \{0,1\}^n} \; (-f(\boldsymbol{x}), |\boldsymbol{x}|) \qquad \text{bi-objective}$$

Algorithm 1 POSS

Input: all variables $V = \{X_1, \ldots, X_n\}$, a given objective f and an integer parameter $k \in [1, n]$
Parameter: the number of iterations T
Output: a subset of V with at most k variables
Process:

```
Let s = {0}^n and P = {s}.
Let t = 0.
while t < T do
    Select s from P uniformly at random.
    Generate s' by flipping each bit of s with prob. 1/n.
    Evaluate f_1(s') and f_2(s').
    if ∄z ∈ P such that z ≺ s' then
        Q = {z ∈ P | s' ⪯ z}.
        P = (P \ Q) ∪ {s'}.
    end if
    t = t + 1.
end while
return arg min_{s∈P,|s|≤k} f_1(s)
```

Initialization: put the special solution $\{0\}^n$ into the population P

Reproduction: pick a solution $\boldsymbol{x}$ randomly from P, and flip each bit of $\boldsymbol{x}$ with prob. $1/n$ to produce a new solution

Updating: if the new solution $\boldsymbol{x}'$ is not dominated by any solution in P, put it into P and delete those solutions weakly dominated by $\boldsymbol{x}'$

Output: select the best feasible solution

Theoretical analysis

POSS can achieve the optimal approximation guarantee, previously obtained by the greedy algorithm

Theorem 1. For monotone approximately submodular maximization with a size constraint, POSS using $\mathrm{E}[T] \le 2eB^2 n$ finds a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \le B$ and

$$\boxed{f(\boldsymbol{x}) \ge (1 - e^{-\gamma}) \cdot \mathrm{OPT}}$$

the optimal polynomial-time approximation ratio
[Das & Kempe, ICML'11; Harshaw et al., ICML'19]

POSS can do better than the greedy algorithm in cases

[Das & Kempe, STOC'08]

Theorem 2. For the Exponential Decay subclass of sparse regression, POSS using $\mathrm{E}[T] = O(B^2(n - B) n \log n)$ finds an optimal solution, while the greedy algorithm cannot

Experiments – sparse regression

the size constraint: $B = 8$ **the number of iterations of POSS: $2eB^2n$**

exhaustive search — greedy algorithms — relaxation methods

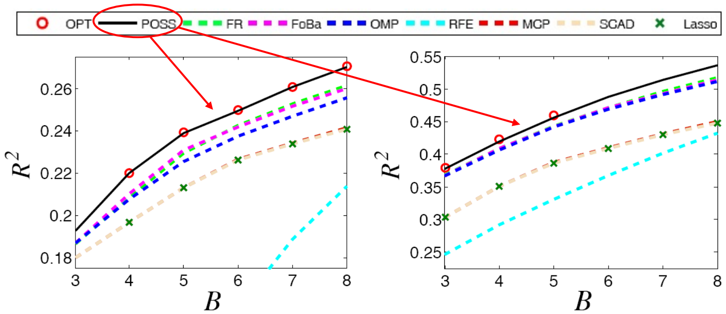
Data set	OPT	POSS	FR	FoBa	OMP	RFE	MCP
housing	.7437±.0297	.7437±.0297	.7429±.0300•	.7423±.0301•	.7415±.0300•	.7388±.0304•	.7354±.0297•
eunite2001	.8484±.0132	.8482±.0132	.8348±.0143•	.8442±.0144•	.8349±.0150•	.8424±.0153•	.8320±.0150•
svmguide3	.2705±.0255	.2701±.0257	.2615±.0260•	.2601±.0279•	.2557±.0270•	.2136±.0325•	.2397±.0237•
ionosphere	.5995±.0326	.5990±.0329	.5920±.0352•	.5929±.0346•	.5921±.0353•	.5832±.0415•	.5740±.0348•
sonar	–	.5365±.0410	.5171±.0440•	.5138±.0432•	.5112±.0425•	.4321±.0636•	.4496±.0482•
triazines	–	.4301±.0603	.4150±.0592•	.4107±.0600•	.4073±.0591•	.3615±.0712•	.3793±.0584•
coil2000	–	.0627±.0076	.0624±.0076•	.0619±.0075•	.0619±.0075•	.0363±.0141•	.0570±.0075•
mushrooms	–	.9912±.0020	.9909±.0021•	.9909±.0022•	.9909±.0022•	.6813±.1294•	.8652±.0474•
clean1	–	.4368±.0300	.4169±.0299•	.4145±.0309•	.4132±.0315•	.1596±.0562•	.3563±.0364•
w5a	–	.3376±.0267	.3319±.0247•	.3341±.0258•	.3313±.0246•	.3342±.0276•	.2694±.0385•
gisette	–	.7265±.0098	.7001±.0116•	.6747±.0145•	.6731±.0134•	.5360±.0318•	.5709±.0123•
farm-ads	–	.4217±.0100	.4196±.0101•	.4170±.0113•	.4170±.0113•	–	.3771±.0110•
POSS: win/tie/loss		–	12/0/0	12/0/0	12/0/0	11/0/0	12/0/0

• denotes that POSS is significantly better by the t-test with confidence level 0.05

POSS is significantly better than all the compared algorithms on all data sets

Experiments – sparse regression

different size constraints: $B = 3 \rightarrow 8$



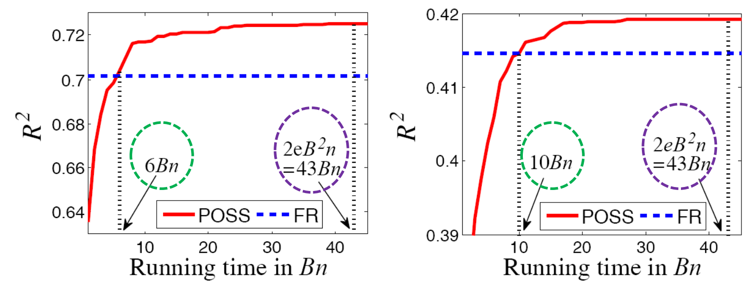
(a) on *svmguide3* (b) on *sonar*

POSS tightly follows OPT, and has a clear advantage over the rest algorithms

Experiments – sparse regression

Running time comparison

OPT: n^B/B^B greedy algorithms (FR): Bn POSS: $2eB^2n$



theoretical running time

(a) on *gisette* (b) on *farm-ads*

POSS can be more efficient in practice

General cost constraints

Original problem

$$max_{X\subseteq V} \quad f(X) \quad s.t. \quad \boxed{|X| \leq B}$$ → size constraints

⇩ extension

$$max_{X\subseteq V} \quad f(X) \quad s.t. \quad \boxed{c(X) \leq B}$$ → general cost constraints

$f(X)$: a monotone approximately submodular objective function

$c(X)$: a monotone approximately submodular cost function

Pareto optimization for general cost constraints

The POMC algorithm [Qian, Shi, Yu and Tang, IJCAI'17]

Transformation:

$$max_{\boldsymbol{x}\in\{0,1\}^n} \; f(\boldsymbol{x}) \quad s.t. \quad c(\boldsymbol{x}) \leq B$$ original

⇩

$$min_{\boldsymbol{x}\in\{0,1\}^n} \; (-f(\boldsymbol{x}), c(\boldsymbol{x}))$$ bi-objective

Algorithm 2 POMC Algorithm

Input: a monotone objective function f, a monotone approximate cost function $\hat{c}$, and a budget B
Parameter: the number T of iterations
Output: a solution $\boldsymbol{x} \in \{0,1\}^n$ with $\hat{c}(\boldsymbol{x}) \leq B$
Process:

```
Let x = {0}^n and P = {x}.
Let t = 0.
while t < T do
    Select x from P uniformly at random.
    Generate x' by flipping each bit of x with prob. 1/n.
    if ∄z ∈ P such that z ≻ x' then
        P = (P \ {z ∈ P | x' ⪰ z}) ∪ {x'}.
    end if
    t = t + 1.
end while
return arg max_{x∈P: ĉ(x)≤B} f(x)
```

Initialization: put the special solution $\{0\}^n$ into the population P

Reproduction: pick a solution x randomly from P, and flip each bit of x with prob. $1/n$ to produce a new solution

Updating: if the new solution $\boldsymbol{x}'$ is not dominated by any solution in P, put it into P and delete those solutions weakly dominated by $\boldsymbol{x}'$

Output: select the best feasible solution

Theoretical analysis

POMC can achieve the best-known approximation guarantee, previously obtained by the generalized greedy algorithm

Theorem 3. For monotone approximately submodular maximization with a general cost constraint, POMC using $\mathrm{E}[T] \leq \boxed{enBP_{max}/\delta_c}$ finds a solution $\boldsymbol{x}$ with $c(\boldsymbol{x}) \leq B$ and

$$\boxed{f(\boldsymbol{x}) \geq \frac{\alpha_f}{2}\left(1 - \frac{1}{e^{\alpha_f}}\right) \cdot \mathrm{OPT}}$$

the best-known polynomial-time approximation ratio
[Zhang & Vorobeychik, AAAI'16]

Proof

Lemma 1. For any $X \subseteq V$, there exists one element $\hat{v} \in V \setminus X$ such that

$$f(X \cup \{\hat{v}\}) - f(X) \geq \boxed{\alpha_f} \frac{c(X \cup \{\hat{v}\}) - c(X)}{B} (\boxed{\mathrm{OPT}} - f(X))$$

submodularity ratio

the optimal function value

Roughly speaking, the improvement on f by adding a specific item is proportional to the current distance to the optimum

Proof

Lemma 1. For any $X \subseteq V$, there exists one item $\hat{v} \in V \setminus X$ such that

$$f(X \cup \{\hat{v}\}) - f(X) \geq \alpha_f \frac{c(X \cup \{\hat{v}\}) - c(X)}{B} (\mathrm{OPT} - f(X))$$

Main idea: a subset

- consider a solution $\boldsymbol{x}$ with $c(\boldsymbol{x}) \leq i \in [0, B)$ and $f(\boldsymbol{x}) \geq \left(1 - \left(1 - \alpha_f \frac{i}{Bk}\right)^k\right) \cdot \mathrm{OPT}$

$i = 0$

$i + c(\boldsymbol{x} \cup \{\hat{v}\}) - c(\boldsymbol{x}) \geq B$

initial solution 00 … 0

$c(00 \ldots 0) = 0$

$f(00 \ldots 0) = 0$

$c(\boldsymbol{x}) < B$

$$f(\boldsymbol{x} \cup \{\hat{v}\}) \geq \left(1 - \left(1 - \alpha_f \frac{i + c(\boldsymbol{x} \cup \{\hat{v}\}) - c(\boldsymbol{x})}{B(k+1)}\right)^{k+1}\right) \cdot \mathrm{OPT}$$
$$\geq \left(1 - \left(1 - \alpha_f \frac{B}{B(k+1)}\right)^{k+1}\right) \cdot \mathrm{OPT}$$
$$\geq (1 - e^{-\alpha_f}) \cdot \mathrm{OPT}$$

Proof

Lemma 1. For any $X \subseteq V$, there exists one item $\hat{v} \in V \setminus X$ such that

$$f(X \cup \{\hat{v}\}) - f(X) \geq \alpha_f \frac{c(X \cup \{\hat{v}\}) - c(X)}{B} (\mathrm{OPT} - f(X))$$

Main idea: a subset

- consider a solution $\boldsymbol{x}$ with $c(\boldsymbol{x}) \leq i \in [0, B)$ and $f(\boldsymbol{x}) \geq \left(1 - \left(1 - \alpha_f \frac{i}{Bk}\right)^k\right) \cdot \mathrm{OPT}$

$i = 0 \xrightarrow{?} i + c(\boldsymbol{x} \cup \{\hat{v}\}) - c(\boldsymbol{x}) \geq B$

initial solution 00 … 0

$c(00 \ldots 0) = 0$

$f(00 \ldots 0) = 0$

$f(\boldsymbol{x} \cup \{\hat{v}\}) \geq (1 - e^{-\alpha_f}) \cdot \mathrm{OPT}$

$f(\boldsymbol{x} \cup \{\hat{v}\}) \leq (f(\boldsymbol{x}) + f(\{\hat{v}\}))/\alpha_f$

$$max\{f(\boldsymbol{x}), f(\{\hat{v}\})\} \geq \frac{\alpha_f}{2}\left(1 - \frac{1}{e^{\alpha_f}}\right) \cdot \mathrm{OPT}$$

the desired approximation guarantee

Proof

Lemma 1. For any $X \subseteq V$, there exists one item $\hat{v} \in V \setminus X$ such that

$$f(X \cup \{\hat{v}\}) - f(X) \geq \alpha_f \frac{c(X \cup \{\hat{v}\}) - c(X)}{B}(\mathrm{OPT} - f(X))$$

Main idea: a subset

- consider a solution $\boldsymbol{x}$ with $c(\boldsymbol{x}) \leq i \in [0, B)$ and $f(\boldsymbol{x}) \geq \left(1 - \left(1 - \alpha_f \frac{i}{Bk}\right)^k\right) \cdot \mathrm{OPT}$
- in each iteration of POMC:
  - select $\boldsymbol{x}$ from the population P
  - flip one specific 0-bit of $\boldsymbol{x}$ to 1-bit (i.e., add the specific item $\hat{v}$ in Lemma 1)

$c(\boldsymbol{x}') \leq i + c(\boldsymbol{x}') - c(\boldsymbol{x})$ and $\boxed{f(\boldsymbol{x}') \geq \left(1 - \left(1 - \alpha_f \frac{i + c(\boldsymbol{x}') - c(\boldsymbol{x})}{B(k+1)}\right)^{k+1}\right) \cdot \mathrm{OPT}}$

Proof

Lemma 1. For any $X \subseteq V$, there exists one item $\hat{v} \in V \setminus X$ such that

$$f(X \cup \{\hat{v}\}) - f(X) \geq \alpha_f \frac{c(X \cup \{\hat{v}\}) - c(X)}{B}(\mathrm{OPT} - f(X))$$

$$f(\boldsymbol{x}') - f(\boldsymbol{x}) \geq \alpha_f \frac{c(\boldsymbol{x}') - c(\boldsymbol{x})}{B} \cdot (\mathrm{OPT} - f(\boldsymbol{x}))$$

$$f(\boldsymbol{x}') \geq \left(1 - \alpha_f \frac{c(\boldsymbol{x}') - c(\boldsymbol{x})}{B}\right) f(\boldsymbol{x}) + \alpha_f \frac{c(\boldsymbol{x}') - c(\boldsymbol{x})}{B} \cdot \mathrm{OPT}$$

$$f(\boldsymbol{x}) \geq \left(1 - \left(1 - \alpha_f \frac{i}{Bk}\right)^k\right) \cdot \mathrm{OPT}$$

$$f(\boldsymbol{x}') \geq \left(1 - \left(1 - \alpha_f \frac{i}{Bk}\right)^k \left(1 - \alpha_f \frac{c(\boldsymbol{x}') - c(\boldsymbol{x})}{B}\right)\right) \cdot \mathrm{OPT} \geq \left(1 - \left(1 - \alpha_f \frac{i + c(\boldsymbol{x}') - c(\boldsymbol{x})}{B(k+1)}\right)^{k+1}\right) \cdot \mathrm{OPT}$$

AM-GM inequality

Proof

Lemma 1. For any $X \subseteq V$, there exists one item $\hat{v} \in V \setminus X$ such that

$$f(X \cup \{\hat{v}\}) - f(X) \geq \alpha_f \frac{c(X \cup \{\hat{v}\}) - c(X)}{B}(\mathrm{OPT} - f(X))$$

Main idea: a subset

- consider a solution $\boldsymbol{x}$ with $c(\boldsymbol{x}) \leq i \in [0, B)$ and $f(\boldsymbol{x}) \geq \left(1 - \left(1 - \alpha_f \frac{i}{Bk}\right)^k\right) \cdot \mathrm{OPT}$
- in each iteration of POMC:
  - select $\boldsymbol{x}$ from the population P the probability: $\frac{1}{|P|}$
  - flip one specific 0-bit of $\boldsymbol{x}$ to 1-bit the probability: $\frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{en}$ (i.e., add the specific item $\hat{v}$ in Lemma 1)

$c(\boldsymbol{x}') \leq i + c(\boldsymbol{x}') - c(\boldsymbol{x})$ and $f(\boldsymbol{x}') \geq \left(1 - \left(1 - \alpha_f \frac{i + c(\boldsymbol{x}') - c(\boldsymbol{x})}{B(k+1)}\right)^{k+1}\right) \cdot \mathrm{OPT}$

$i \longrightarrow i + c(\boldsymbol{x}') - c(\boldsymbol{x}) \geq i + \delta_c$ the probability: $\frac{1}{|P|} \cdot \frac{1}{en}$

$min\{c(\boldsymbol{x} \cup \{v\}) - c(\boldsymbol{x}) \mid v \notin \boldsymbol{x}\}$

Proof

Lemma 1. For any $X \subseteq V$, there exists one item $\hat{v} \in V \setminus X$ such that

$$f(X \cup \{\hat{v}\}) - f(X) \geq \alpha_f \frac{c(X \cup \{\hat{v}\}) - c(X)}{B}(\mathrm{OPT} - f(X))$$

Main idea: a subset

- consider a solution $\boldsymbol{x}$ with $c(\boldsymbol{x}) \leq i \in [0, B)$ and $f(\boldsymbol{x}) \geq \left(1 - \left(1 - \alpha_f \frac{i}{Bk}\right)^k\right) \cdot \mathrm{OPT}$
- in each iteration of POMC:

$i \longrightarrow i + \delta_c$ the probability: $\boxed{\frac{1}{|P|} \cdot \frac{1}{en}} \xrightarrow{|P| \leq P_{max}} \frac{1}{eP_{max}n}$

$i \longrightarrow i + \delta_c$ the expected number of iterations: $eP_{max}n$

$i = 0 \longrightarrow i + c(\boldsymbol{x} \cup \{\hat{v}\}) - c(\boldsymbol{x}) \geq B$

the expected number of iterations: $\frac{B}{\delta_c} \cdot eP_{max}n$

Theoretical analysis

POMC can achieve the best-known approximation guarantee, previously obtained by the generalized greedy algorithm

Theorem 3. [Qian, Shi, Yu and Tang, IJCAI'17] For monotone approximately submodular maximization with a general cost constraint, POMC using E[T] $\leq \boxed{enBP_{max}/\delta_c}$ finds a solution $\boldsymbol{x}$ with $c(\boldsymbol{x}) \leq B$ and

$$\boxed{f(\boldsymbol{x}) \geq \frac{\alpha_f}{2}\left(1 - \frac{1}{e^{\alpha_f}}\right) \cdot \mathrm{OPT}}$$

the best-known polynomial-time approximation ratio [Zhang & Vorobeychik, AAAI'16]

By limiting the largest population size P_{max}, we get the EAMC algorithm whose running time is polynomial

Theorem 4. [Bian, Feng, Qian and Yu, AAAI'20] For monotone approximately submodular maximization with a general cost constraint, EAMC using E[T] $\leq \boxed{2en^2(n+1)}$ finds a solution $\boldsymbol{x}$ with $c(\boldsymbol{x}) \leq B$ and

$$\boxed{f(\boldsymbol{x}) \geq \frac{\alpha_f}{2}\left(1 - \frac{1}{e^{\alpha_f}}\right) \cdot \mathrm{OPT}}$$

Experiments – sensor placement

- Sensor placement [Krause et al., JMLR'08]: select a subset of locations to install sensors such that the entropy is maximized

Formally stated: given n locations $V = \{v_1, \dots, v_n\}$ and a budget B, let o_j denote the observation variable by installing a sensor at v_j, and then

$$max_{X \subseteq V} \quad H(\{o_j \mid v_j \in X\}) \quad s.t. \quad c(X) \leq B$$

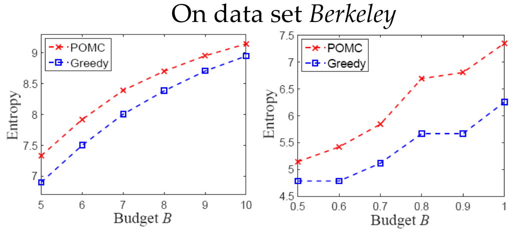
- Constraints: cardinality $|X| \leq B \in \{5, \dots, 10\}$ and routing $c(X) \leq B \in \{0.5, \dots, 1\}$

  the shortest walk to visit each node in X at least once

- Data sets: *Berkeley* ($n = 55$), *Beijing* ($n = 36$)
- For POMC on each data set with each B value, the run is repeated for 10 runs independently, and the average results are reported
- Compare POMC with the generalized greedy algorithm

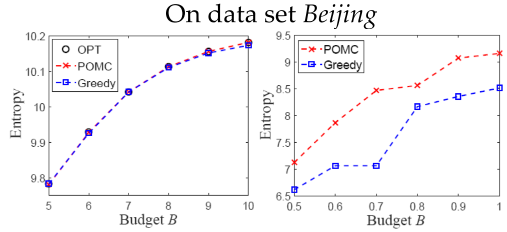
Experiments – sensor placement

On data set *Berkeley*



(a) (Berkeley, cardinality) (b) (Berkeley, routing)

On data set *Beijing*



(c) (Beijing, cardinality) (d) (Beijing, routing)

POMC is better in most cases, and never worse

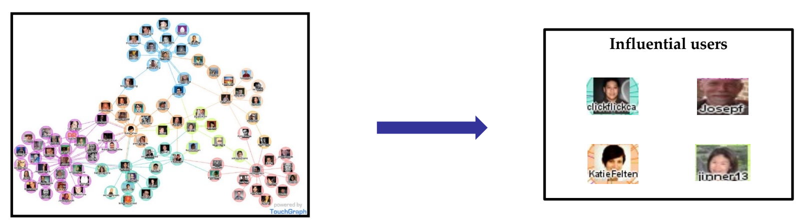
Experiments – influence maximization

- Influence maximization [Kempe et al., KDD'03]: select a subset of users from a social network such that the influence spread is maximized

Formally stated: given a directed graph $G = (V, E)$ with $|V| = n$, edge probabilities $p_{u,v}$ $((u, v) \in E)$ and a budget B, then

$$max_{X \subseteq V} \quad f(X) \quad s.t. \quad c(X) \leq B$$

The expected number of nodes activated by propagating from X



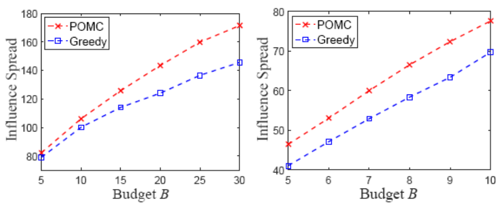
Experiments – influence maximization

- Influence maximization [Kempe et al., KDD'03]: select a subset of users from a social network such that the influence spread is maximized

Formally stated: given a directed graph $G = (V, E)$ with $|V| = n$, edge probabilities $p_{u,v}$ $((u, v) \in E)$ and a budget B, then

$$max_{X \subseteq V} \quad f(X) \quad s.t. \quad c(X) \leq B$$

The expected number of nodes activated by propagating from X



(a) (Digg, cardinality) (b) (Synthetic, routing)

POMC is always better

Pareto optimization vs. Greedy algorithm

(Generalized) Greedy algorithm:

- Generate a new solution by adding a single item (i.e., single-bit forward search: 0 → 1)
- Keep only one solution

Pareto optimization:

- Generate a new solution by flipping each bit with prob. $1/n$
  - ➢ single-bit forward search : 0 → 1
  - ➢ backward search : 1 → 0
  - ➢ multi-bit search : 00 → 11
- Keep a set of non-dominated solutions due to bi-objective optimization

Pareto optimization may have a better ability of escaping from local optima

Problems with Dynamic Constraints

[V. Roostapour, A Neumann, F. Neumann, T. Friedrich: Pareto Optimization for Subset Selection with Dynamic Cost Constraints, AAAI'19]

Dynamic Constraints

- Many real world optimization problems are dynamic and/or stochastic.
- Often the goal function to be optimized is fixed (reduce cost / maximize profit).
- Resources to achieve these goal are usually changing.

- Example:
- ➢ Trucks/trains may break down and/or be repaired.
- ➢ Algorithms have to react to such changes that effect the constraints of the given problem.

Now:

- Study of (adaptive) greedy algorithms and Pareto optimization approaches for problems with dynamic constraints.

Definitions

The Static Problem [C. Qian, J. Shi, Y. Yu, K. Tang, IJCAI'17]

Given a monotone objective function $f : 2^V \rightarrow \mathbb{R}^+$, the monotone cost function $c : 2^V \rightarrow \mathbb{R}^+$ and budget B, the aim is to find X such that

$$\boxed{X = \arg\max_{Y \subseteq V} f(Y) \text{ s.t. } c(Y) \leq B.}$$

$$\boxed{\phi} = (\alpha_f/2)(1 - \frac{1}{e^{\alpha_f}})$$

ϕ - approximation

The Dynamic Problem

[V. Roostapour, A. Neumann, F. Neumann, T. Friedrich, AAAI'19]

Let X be a ϕ-approximation for the static problem. The dynamic problem is given by a sequence of changes where in each change the current budget B changes to $B^* = B + d$, $d \in \mathbb{R}_{\geq -B}$. The goal is to compute a ϕ-approximation X' for each newly given budget B^*.

Greedy Algorithms

Algorithm 1: Generalized Greedy Algorithm

input: Initial budget constraint B.

$X \leftarrow \emptyset$;
$V' \leftarrow V$;
repeat
  $v^* \leftarrow \arg\max_{v \in V'} \frac{f(X \cup v) - f(X)}{\hat{c}(X \cup v) - \hat{c}(X)}$;
  if $\hat{c}(X \cup v^*) \leq B$ **then**
    $X \leftarrow X \cup v^*$;
  $V' \leftarrow V' \setminus \{v^*\}$;
until $V' \leftarrow \emptyset$;
$v^* \leftarrow \arg\max_{v \in V; \hat{c}(v) \leq B} f(v)$;
return $\arg\max_{S \in \{X, v^*\}} f(S)$;

[Zhang and Vorobeychik, AAAI' 16]

$$K_c = \max\{|X| : c(X) \leq B\}$$

$$\tilde{X}_B = \arg\max\{f(X) \mid c(X) \leq \alpha_c \frac{B(1+\alpha_c^2(K_c-1)(1-\kappa_c)))}{\psi K_c}\}$$

$\boxed{(1/2)(1-\frac{1}{e})\text{-approximate solution}}$

Algorithm 2: Adaptive Generalized Greedy Algorithm

input: Initial solution X, Budget constraint B, New budget constraint B^*.

if $B^* < B$ **then**
  while $\hat{c}(X) > B^*$ **do**
    $v^* \leftarrow \arg\min_{v \in X} \frac{f(X) - f(X \setminus \{v\})}{\hat{c}(X) - \hat{c}(X \setminus \{v\})}$;
    $X \leftarrow X \setminus \{v^*\}$;
else if $B^* > B$ **then**
  $V' \leftarrow V \setminus X$;
  repeat
    $v^* \leftarrow \arg\max_{v \in V'} \frac{f(X \cup v) - f(X)}{\hat{c}(X \cup v) - \hat{c}(X)}$;
    if $\hat{c}(X \cup v^*) \leq B^*$ **then**
      $X \leftarrow X \cup v^*$;
    $V' \leftarrow V' \setminus \{v^*\}$;
  until $V' \leftarrow \emptyset$;
$v^* \leftarrow \arg\max_{v \in V; \hat{c}(v) \leq B^*} f(v)$;
return $\arg\max_{S \in \{X, v^*\}} f(S)$;

[V. Roostapour, A Neumann, F. Neumann, T. Friedrich, AAAI'19]

The adaptive generalized greedy algorithm can not deal with dynamic increases of the constraint bound.

Approximation Adaptive Greedy

Consider n items $e_i = (c_i, f_i), 1 \leq i \leq n+1$

- Low profit items: $e_i = (1, \frac{1}{n}),\ 1 \leq i \leq n/2$
- High profit items: $e_i = (2, 1),\ n/2+1 \leq i \leq n$
- Special item: $e_{n+1} = (1, 3)$

Linear objective and constraint function:

$$\boxed{f_{inc}(X) = \sum_{e_i \in X} f_i \qquad c_{inc}(X) = \sum_{e_i \in X} c_i}$$

Consider the following dynamic schedule:

- Start with B = 1 and increase B by 1 in each of n/2 steps.

Approximation Adaptive Greedy

Theorem 3. *Given the dynamic knapsack problem (f_{inc}, c_{inc}). Starting with $B = 1$ and increasing the bound $n/2$ times by 1, the adaptive greedy algorithm computes a solution that has approximation ratio $O(1/n)$.*

[V. Roostapour, A. Neumann, F. Neumann, T. Friedrich, AAAI'19]

Proof idea:

- For B=1, the special item $\boxed{e_{n+1} = (1, 3)}$ is included.
- During n/2 steps increasing the budget by 1, all low profit items are included.
- For the obtained set S, we have

$$\boxed{f(S) = 3 + (n/2) \cdot (1/n) = 7/2 \text{ and } c(S) = 1 + n/2}$$

- Optimal set S* consists of special item and n/4 high profit items and we have $\boxed{f(S^*) = 3 + \frac{n}{4}}$
- Approximation ratio $\boxed{(7/2)/(3 + n/4) = O(1/n)}$

Pareto Optimization

Algorithm 3: POMC Algorithm

input: Initial budget constraint B, time T

$X \leftarrow \{0\}^n$;
Compute $(f_1(X), f_2(X))$;
$P \leftarrow \{x\}$;
$t \leftarrow 0$;
while $t < T$ **do**
  Select X from P uniformly at random;
  $X' \leftarrow$ flip each bit of X with probability $\frac{1}{n}$;
  Compute $(f_1(X'), f_2(X'))$;
  if $\nexists Z \in P$ *such that* $Z \succ X'$ **then**
    $\boxed{P \leftarrow (P \setminus \{Z \in P \mid X' \succeq Z\}) \cup \{X'\}}$;
  $t = t + 1$;
return $\arg\max_{X \in P : \hat{c}(X) \le B} f(x)$

$$\boxed{\arg\max X \in \{0,1\}^n (f_1(X), f_2(X))}$$

where $f_1(X) = \begin{cases} -\infty, & \hat{c}(X) > B + 1 \\ f(X), & \text{otherwise} \end{cases}$, $f_2(X) = -\hat{c}(X)$.

Theoretical Results POMC

Theorem 5. *Starting from $\{0\}^n$, POMC computes for any budget $b \in [0, B]$ a $\phi = (\alpha_f/2)(1 - 1/e^{\alpha_f})$-approximate solution after $T = cnP_{max} \cdot \frac{B}{\delta_{\hat{c}}}$ iterations with the constant probability, where $c \ge 8e+1$ is a sufficiently large arbitrary constant.*

Theorem 6. *Let POMC has population P such that for every budget $b \in [0, B]$, there is a ϕ-approximation in P. After changing the budget to $B^* > B$, POMC has computed within $T = cnP_{max} \frac{d}{\delta_{\hat{c}}}$ steps for every $b \in [0, B^*]$ a ϕ-approximation with probability $\Omega(1)$.*

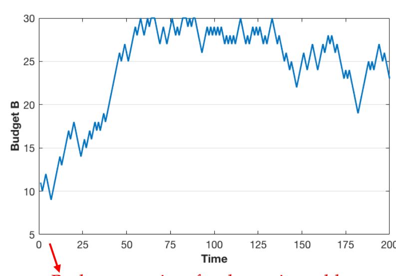
[V. Roostapour, A. Neumann, F. Neumann, T. Friedrich, AAAI'19]

Experiments

We consider the influence maximization problem in social networks. [Zhang & Vorobeychik, AAAI'16]

Two types of constraints:

- Routing constraint on routing cost for selected users
- Cardinality constraint on number of selected users



Budget over time for dynamic problems

For both problems, we vary the constraint bound B over time.

- POMC has τ = 1000, 5000, 10000 iterations after every change to recompute good solution.

Experimental results

Dynamic routing constraints

Changes	GGA		AGGA		$POMC_{1000}$		$POMC_{5000}$		$POMC_{10000}$	
	mean	st	mean	st	mean	st	mean	st	mean	st
1-25	85.0349	12.88	81.5734	14.07	66.3992	17.95	77.8569	18.76	86.1057	17.22
26-50	100.7344	22.16	96.1386	23.99	104.9102	15.50	117.6439	16.71	122.5604	15.54
51-75	118.1568	30.82	110.4893	29.50	141.8249	5.64	155.2126	5.08	158.7228	5.20
76-100	127.3422	31.14	115.2978	27.66	149.0259	3.36	159.9100	3.28	162.7353	3.65
101-125	132.3502	29.62	116.9768	25.45	150.3415	3.17	160.1367	2.81	161.2852	2.68
126-150	134.5256	27.69	118.6962	24.19	147.8998	7.36	154.7319	8.77	154.1470	7.43
151-175	135.7651	25.89	119.4982	22.85	147.2478	4.68	153.1417	5.32	151.2966	3.17
176-200	135.5133	24.41	119.1491	22.04	139.5072	8.08	143.6928	9.16	143.9832	8.67

Dynamic cardinality constraints

Changes	GGA		AGGA		$POMC_{1000}$		$POMC_{5000}$		$POMC_{10000}$	
	mean	st	mean	st	mean	st	mean	st	mean	st
1-25	130.9410	14.71	130.6550	14.36	84.8898	24.32	114.8272	23.09	121.1330	19.72
26-50	145.6766	20.70	145.0774	20.11	133.2130	14.69	155.4231	13.98	158.0245	14.34
51-75	160.2780	26.86	159.6331	26.50	164.9157	3.84	184.3274	3.45	187.1952	3.68
76-100	167.9512	26.84	167.3365	26.60	171.5600	1.89	189.4834	2.74	189.6107	2.78
101-125	172.1483	25.45	171.6884	25.35	174.3528	2.11	188.2120	2.32	188.7572	2.46
126-150	174.0582	23.77	173.6528	23.72	174.0404	5.88	183.0188	6.65	183.8033	6.47
151-175	175.1998	22.23	174.8330	22.21	174.5846	4.03	181.3669	4.01	188.4192	3.60
176-200	175.1023	20.94	174.7836	20.92	168.8791	8.05	173.8794	7.28	175.2773	7.23

Summary

- Dynamic problems play a key role in the area of optimization.
- We have shown that an adaptive version of the generalized greedy algorithm only achieves arbitrary bad performance for simple submodular problems.
- The POMC Pareto optimization approach caters for dynamic changes by having for each possible budget $b \leq B$ a good approximation.
- POMC can recompute good approximations for all new possible budgets in the case of budget $b \in [B, B^*]$ increase from B to B^* efficiently.
- Experiments on influence maximization in social networks show the advantage of POMC over greedy approaches.

Problems with Chance Constraints

[B. Doer, C. Doerr, A. Neumann, F. Neumann, A. M. Sutton: Optimization of Chance-Constrained Submodular Functions, AAAI'20]

Chance Constraints - Motivation

- Often problems involve stochastic components and constraints that can only be violated with a small probability.
- We investigate submodular problems with chance constraints and show that the adaptation of simple greedy algorithms asymptotically only looses a factor of $\boxed{1-o(1)}$ in terms of the worst case approximation obtained.

Chance Constraints

Let S be a potential solution to a given submodular problem, W(S) be its random weight and B be a given weight bound.
We consider chance constraints of the form:

$$\boxed{\Pr[W(S) > B] \leq \alpha.}$$

small, e.g. 0.001

Weight bound can only be violated with a small probability.

Setting for Random Weights

- We consider two settings for random weights of a given set of items.
- Both settings assume that the weights of the items are chosen independent of each other.

Uniform independent and identically distributed (IID) weights:

$$\boxed{W(s) \in [a-\delta, a+\delta] \ (\delta \leq a)}$$

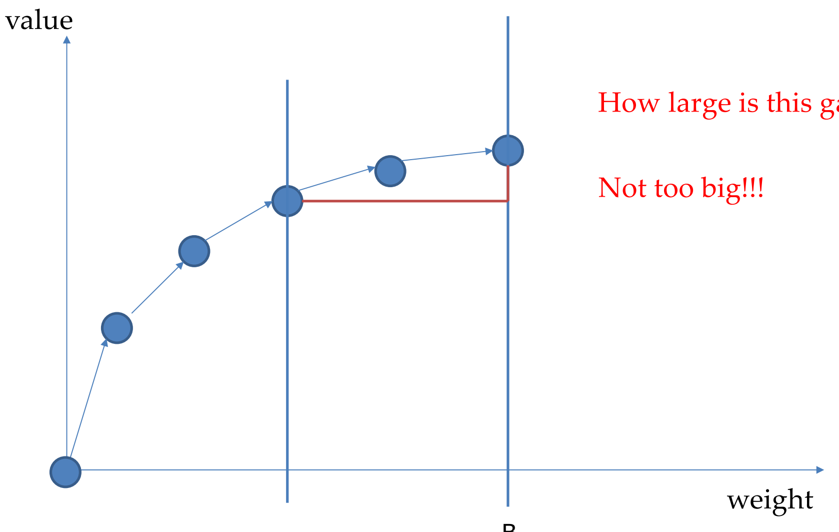
Uniform Weights with same dispersion

$$\boxed{W(s) \in [a(s)-\delta, a(s)+\delta]}$$

Chance Constraints

- One of the difficulties lies in evaluating whether a given solution fulfills the chance constraint.
- Use surrogate functions such as Chernoff bounds and Chebyshev's inequality to determine whether a solution is feasible. [Chebyshev, MPA'67; Chernoff, AMS'52]
- These bounds don't allow for a precise calculation for the probability of a constraint violation.
- However, the give an upper bound and a solution is accepted if its upper bound is at most α .
- For our settings, we establish conditions based on the difference in expected weight and constraint B that show when a given solution is feasible.

Greedy Algorithms



How large is this gap?

Not too big!!!

Chance constraint case stops earlier.

Chance Constraint Conditions

Chernoff:

Lemma 1. *Let $W(s) \in [a(s)-\delta, a(s)+\delta]$ be independently chosen uniformly at random. If*

$$(B - E[W(X)]) \geq \sqrt{3\delta k \ln(1/\alpha)},$$

where $k = |X|$, then $\Pr[W(X) > B] \leq \alpha$.

Chebyshev:

Lemma 2. *Let X be a solution with expected weight $E[W(X)]$ and variance $\mathrm{Var}[W(X)]$. If*

$$B - E[W(X)] \geq \sqrt{\frac{(1-\alpha)\,\mathrm{Var}[W(X)]}{\alpha}}$$

then $\Pr[W(X) > B] \leq \alpha$.

Uniform IID Weights

Greedy Algorithm

Algorithm 1: Greedy Algorithm (GA)

input: Set of elements V, budget constraint B, failure probability α.

$S \leftarrow \emptyset$;
$V' \leftarrow V$;
repeat
  $v^* \leftarrow \arg\max_{v \in V'} (f(S \cup \{v\}) - f(S))$;
  if $\widehat{\Pr}[W(S \cup \{v^*\}) > B] \leq \alpha$ **then**
    $S \leftarrow S \cup \{v^*\}$;
  $V' \leftarrow V' \setminus \{v^*\}$;
until $V' \leftarrow \emptyset$;
return S;

Theorem: If B = ω(1) then GA gives a (1-o(1))(1-1/e)- approximation for each monotone submodular function when using Chernoff or Chebyshev for the chance constraint evaluation.

Experiments

We consider the influence maximization problem in social networks. [Zhang & Vorobeychik, AAA'16; Leskovec et al., SIGKDD'07; Kempe et al., SIGKDD'03; Kempe et al., TC'15]

- Given a graph G = (V,E) where nodes are users and and edge (u,v) have probability weights which determines how likely user u influences user v.
- Expected influence score is computed by propagation from the set of selected users. This is done through a simulation.
- In addition there is a constraint on the cost of selecting users.

- Goal: select a set of users that maximizes influence under the given constraint.
- Chance constraint settings: expected weights of 1 for IID case.

Experimental Results – Cost values

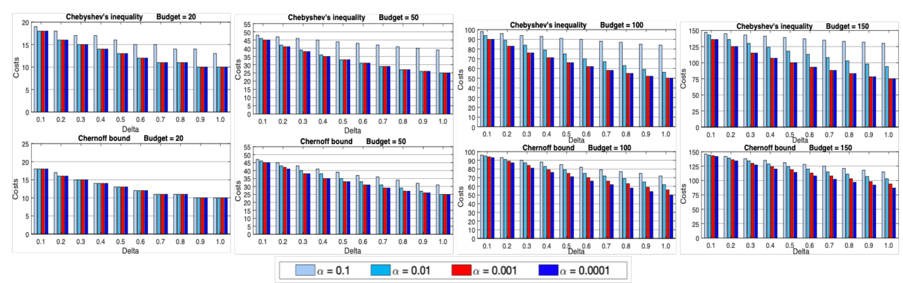


Figure 2: Maximal cost values for budgets $B = 20, 50, 100, 150$ (from left to right) using Chebyshev's inequality (top) and Chernoff bound (bottom) for $\alpha = 0.1, 0.01, 0.001, 0.0001$ with uniform expected weights set to 1.

Experimental Results – Function values

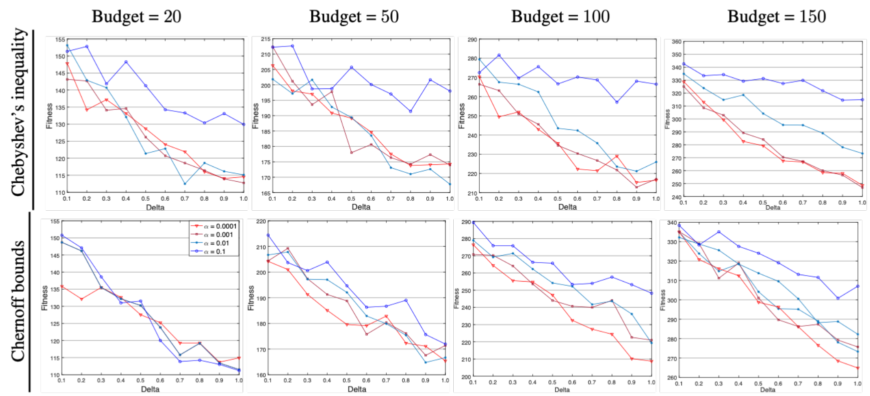


Figure 1: Function value for budgets $B = 20, 50, 100, 150$ (from left to right) using Chebyshev's inequality (top) and Chernoff bound (bottom) for $\alpha = 0.1, 0.01, 0.001, 0.0001$ with all the expected weights 1.

Uniform Weights with same dispersion

Generalized Greedy Algorithm

Algorithm 2: Generalized Greedy Algorithm (GGA)

input: Set of elements V, budget constraint B, failure probability α.

$S \leftarrow \emptyset$;
$V' \leftarrow V$;
repeat
  $v^* \leftarrow \arg\max_{v \in V'} \frac{f(S \cup \{v\}) - f(S)}{E[W(S \cup \{v\}) - W(S)]}$;
  if $\Pr[W(S \cup \{v^*\}) > B] \leq \alpha$ **then**
    $S \leftarrow S \cup \{v^*\}$;
  $V' \leftarrow V' \setminus \{v^*\}$;
until $V' \leftarrow \emptyset$;
$v^* \leftarrow \arg\max_{\{v \in V; \Pr[W(v) > B] \leq \alpha\}} f(v)$;
return $\arg\max_{Y \in \{S, \{v^*\}\}} f(Y)$;

Theorem: If B = ω(1) then GGA gives a (1/2-o(1))(1-1/e)-approximation for each monotone submodular function when using Chernoff or Chebyshev for the chance constraint evaluation.

Experimental Results – Function values

Figure 3: Function values for budgets $B = 100$ (left) and $B = 500$ (right) using Chebyshev's inequality and Chernoff bound for $\alpha = 0.1, 0.01, 0.001, 0.0001$ with degree dependent random weights.

Expected weight 1+degree(v) for uniform with same dispersion case.

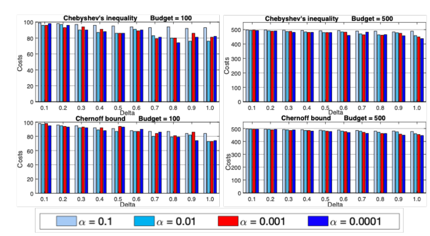


Figure 4: Maximal cost values for budget $B = 100$ (left) and $B = 500$ using Chebyshev's inequality (top) and Chernoff bound (bottom) for $\alpha = 0.1, 0.01, 0.001, 0.0001$ with degree dependent random weights.

Summary

- Optimization problems often involve stochastic components that effect the constraints of the problem.
- We presented a (first) study on submodular functions with chance constraints.
- We showed that simple greedy algorithms popular for dealing with monotone submodular functions can be easily adapted to the chance constrained case.
- In terms of approximation, we asymptotically only loose a factor of 1-o(1).
- Experimental results show the change in solution quality dependent on the uncertainty of the weights and the chance constraint violation probability.

Problems with Chance Constraints: Evolutionary Multi-Objective Algorithms

[A. Neumann and F. Neumann: Optimising Monotone Chance-Constrained Submodular Functions Using Evolutionary Multi-Objective Algorithms, PPSN'20]

Problem Definition

- We consider the performance of the Global Simple Evolutionary Multi-Objective Optimizer (GSEMO) and Non-dominated Sorting Genetic Algorithm (NSGA-II) for the optimisation of chance constrained submodular functions.
  [Doerr, B., Doerr, C., Neumann, A., Neumann, F., Sutton, A. M., AAAI'20; Xie, Neumann, A., Neumann, F., GECCO'20; Xie et al., GECCO 2019; Assimi et al., ECAI'20]
- Use and evaluate Pr using Chernoff bounds or Chebyshev's inequality.

$$\boxed{\Pr(W(X) > B) \leq \hat{\Pr}(W(X) > B)}$$

- Uniform IID weights:

$$\boxed{W(s) \in [a - \delta, a + \delta] \ (\delta \leq a).}$$

- Uniform weights with same dispersion:

$$\boxed{W(s) \in [a(s) - \delta, a(s) + \delta].}$$

Algorithm

Global Simple Evolutionary Multi-Objective Optimizer [Giel & Wegener, STACS'03]

Algorithm 1: Global SEMO

```
Choose x ∈ {0,1}^n uniformly at random;
P ← {x};
repeat
    Choose x ∈ P uniformly at random;
    Create y by flipping each bit x_i of x with probability 1/n;
    if ∄w ∈ P : w ≻ y then
        S ← (P ∪ {y}) \ {z ∈ P | y ≽ z};
until stop;
```

Multi-Objective Formulation

[Motwani & Raghavan,'95; Doerr & Neumann, NCS '20;
Xie, Harper, Assimi, Neumann, A., Neumann, F., GECCO'19]

Uniform IID Weights:

$$g(X) = (g_1(X), g_2(X))$$

$$g_1(X) = \begin{cases} E_W(X) - C & \text{if} \quad (C - E_W(X))/(\delta \cdot |X|) \geq 1 \\ \hat{P}r(W(X) > C) & \text{if } (E_W(X) < C) \wedge (C - E_W(X))/(\delta |X| < 1) \\ 1 + (E_W(X) - C) & \text{if} \quad E_W(X) \geq C \end{cases}$$

$$g_2(X) = \begin{cases} f(X) & \text{if} \quad g_1(X) \leq \alpha \\ -1 & \text{if } \hat{P}r(W(X) > C) > \alpha \end{cases}$$

Uniform Weights with the Same Dispersion:

$$\hat{g}(X) = (\hat{g}_1(X), g_2(X))$$

$$\hat{g}_1(X) = E_W(X)$$

Theoretical Results

Uniform IID Weights:

Theorem: Let k = min{n + 1, ⌊C/a⌋} and assume ⌊C/a⌋ = ω(1). Then the expected time until GSEMO has computed a (1−o(1))(1−1/e)-approximation for a given monotone submodular function under a chance constraint with uniform iid weights is O(nk(k + log n)).

Uniform Weights with the Same Dispersion:

Theorem: If $C/a_{max} = \omega(1)$ then GSEMO obtains a (1/2 − o(1))(1 − 1/e)-approximation for a given monotone submodular function under a chance constraint with uniform weights having the same dispersion in expected time $O(P_{max} \cdot n(C/a_{min} + \log n + \log(a_{max}/a_{min})))$.

[A. Neumann and F. Neumann, PPSN'20]

Experimental Results

Results for Influence Maximization with uniform chance constraints.

[Kempe et al., SIGKDD '03]

C	α	δ	GA (1)	GSEMO (2)					NSGA-II (3)				
				mean	min	max	std	stat	mean	min	max	std	stat
20	0.1	0.5	51.51	**55.75**	54.44	56.85	0.5571	$1^{(+)}$	55.66	54.06	56.47	0.5661	$1^{(+)}$
	0.1	1.0	46.80	**50.65**	49.53	51.68	0.5704	$1^{(+)}$	50.54	49.61	52.01	0.6494	$1^{(+)}$
50	0.1	0.5	90.55	**94.54**	93.41	95.61	0.5390	$1^{(+)}, 3^{(+)}$	92.90	90.75	94.82	1.0445	$1^{(+)}, 2^{(-)}$
	0.1	1.0	85.71	**88.63**	86.66	90.68	0.9010	$1^{(+)}, 3^{(+)}$	86.89	85.79	88.83	0.8479	$1^{(+)}, 2^{(-)}$
100	0.1	0.5	144.16	**147.28**	145.94	149.33	0.8830	$1^{(+)}, 3^{(+)}$	144.17	142.37	146.18	0.9902	$2^{(-)}$
	0.1	1.0	135.61	**140.02**	138.65	142.52	0.7362	$1^{(+)}, 3^{(+)}$	136.58	134.80	138.21	0.9813	$2^{(-)}$
20	0.001	0.5	48.19	**50.64**	49.10	51.74	0.6765	$1^{(+)}$	50.33	49.16	51.25	0.5762	$1^{(+)}$
	0.001	1.0	39.50	**44.53**	43.63	45.55	0.4687	$1^{(+)}$	44.06	42.18	45.39	0.7846	$1^{(+)}$
50	0.001	0.5	75.71	**80.65**	78.92	82.19	0.7731	$1^{(+)}$	80.58	79.29	81.63	0.6167	$1^{(+)}$
	0.001	1.0	64.49	69.79	68.89	71.74	0.6063	$1^{(+)}$	**69.96**	68.90	71.05	0.6192	$1^{(+)}$
100	0.001	0.5	116.05	**130.19**	128.59	131.51	0.7389	$1^{(+)}, 3^{(+)}$	127.50	125.38	129.74	0.9257	$1^{(+)}, 2^{(-)}$
	0.001	1.0	96.18	**108.95**	107.26	109.93	0.6466	$1^{(+)}, 3^{(+)}$	107.91	106.67	110.17	0.7928	$1^{(+)}, 2^{(-)}$

Experimental Results

Results for Maximum Coverage with uniform chance constraints.

[Feige, ACM'98, Khuller et al., IPL'99]

C	α	δ	GA (1)	GSEMO (2)					NSGA-II (3)				
				mean	min	max	std	stat	mean	min	max	std	stat
10	0.1	0.5	448.00	**458.80**	451.00	461.00	3.3156	$1^{(+)}$	457.97	449.00	461.00	4.1480	$1^{(+)}$
	0.1	1.0	376.00	**383.33**	379.00	384.00	1.7555	$1^{(+)}$	382.90	379.00	384.00	2.0060	$1^{(+)}$
15	0.1	0.5	559.00	**559.33**	555.00	562.00	2.0057	$3^{(+)}$	557.23	551.00	561.00	2.4309	$1^{(-)}, 2^{(-)}$
	0.1	1.0	503.00	**507.80**	503.00	509.00	1.1567	$1^{(+)}$	507.23	502.00	509.00	1.8323	$1^{(+)}$
20	0.1	0.5	587.00	**587.20**	585.00	589.00	1.2149	$3^{(+)}$	583.90	580.00	588.00	1.9360	$1^{(-)}, 2^{(-)}$
	0.1	1.0	569.00	**569.13**	566.00	572.00	1.4559	$3^{(+)}$	565.30	560.00	569.00	2.1520	$1^{(-)}, 2^{(-)}$
10	0.001	0.5	413.00	**423.67**	418.00	425.00	1.8815	$1^{(+)}$	422.27	416.00	425.00	2.6121	$1^{(+)}$
	0.001	1.0	376.00	**383.70**	379.00	384.00	1.1492	$1^{(+)}$	381.73	377.00	384.00	2.6514	$1^{(+)}$
15	0.001	0.5	526.00	**527.97**	525.00	532.00	2.1573	$1^{(+)}$	527.30	520.00	532.00	2.7436	
	0.001	1.0	448.00	**458.87**	453.00	461.00	2.9564	$1^{(+)}$	457.10	449.00	461.00	4.1469	$1^{(+)}$
20	0.001	0.5	568.00	**568.87**	565.00	572.00	1.5025	$3^{(+)}$	564.60	560.00	570.00	2.7618	$1^{(-)}, 2^{(-)}$
	0.001	1.0	526.00	**528.03**	525.00	530.00	1.8843	$1^{(+)}$	527.07	522.00	530.00	2.2427	

Summary

- We presented a first runtime analysis of evolutionary algorithms for the optimisation of submodular functions with chance constraints.
- We showed that GSEMO using a multi-objective formulation of the problem based on tail inequalities is able to achieve the same approximation guarantee as recently studied greedy approaches.
- Experimental results show that GSEMO computes significantly better solutions than the greedy approach and often outperforms NSGA-II.

Summary

- Many real-world optimisation problems can be formulated in terms of optimising a submodular function under a given set of constraints.
- A wide range of state-of-the-art results for submodular problems have been obtained through evolutionary computing techniques.
- Bi-objective formulations of constrained submodular optimisation problems in terms of Pareto optimisation enable evolutionary algorithms to achieve
  - best theoretical performance guarantees and
  - state-of-the-art practical results

  for a wide range of submodular optimisation problems.
- These approaches are also able to deal with dynamic and stochastic constraints in a very efficient way.

References

- L. Lovász. Submodular functions and convexity. In: *A Bachem et al. (Eds): Mathematical Programming The State of the Art*, 235-257, 1983.
- G. L. Nemhauser, L. A. Wolsey and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming*, 1978, 14(1): 265–294.
- J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Forty-first Annual ACM Symposium on Theory of Computing (STOC)*, pages 323–332, 2009.
- T. Friedrich, F. Neumann: Maximizing Submodular functions under matroid vonstraints by evolutionary algorithms. *Evolutionary Computation 23(4)*, MIT Press, 543-558, 2015.
- D. Kempe, J. Kleinberg and E. Tardos. Maximizing the spread of influence through a social network. In: *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pp. 137–146, Washington, DC, 2003.
- J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 2004, 50(10): 2231–2242.
- A. Das and D. Kempe. Algorithms for subset selection in linear regression. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'08)*, pp. 45–54, Victoria, Canada, 2008.

References

- A. Krause, A. Singh and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 2008, 9: 235–284.
- A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In: *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pp. 1057–1064, Bellevue, WA, 2011.
- C. Qian, Y. Yu and Z.-H. Zhou. Subset selection by Pareto optimization. In: *Advances in Neural Information Processing Systems 28 (NIPS'15)*, pp.1765-1773, Montreal, Canada, 2015.
- H. Zhang and Y. Vorobeychik. Submodular optimization with routing constraints. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*, pp. 819–826, Phoenix, AZ, 2016.
- C. Qian, J.-C. Shi, Y. Yu and K. Tang. On subset selection with general cost constraints. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, pp.2613-2619, Melbourne, Australia, 2017.
- A. A. Bian, J. M. Buhmann, A. Krause and S. Tschiatschek. Guarantees for greedy maximization of non-submodular functions with applications. In: *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, pp. 498–507, Sydney, Australia, 2017.

References

- E. R. Elenberg, R. Khanna, A. G. Dimakis and S. Negahban. Restricted strong convexity implies weak submodularity. *Annals of Statistics*, 2018, 46(6B): 3539–3568.
- C. Qian, Y. Yu and K. Tang. Approximation guarantees of stochastic greedy algorithms for subset selection. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18),* pp. 1478–1484, Stockholm, Sweden, 2018.
- C. Qian, Y. Yu, K. Tang, X. Yao and Z.-H. Zhou. Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artificial Intelligence,* 2019, 275: 279-294.
- C. Harshaw, M. Feldman, J. Ward and A. Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In: *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, pp. 2634–2643, Long Beach, CA, 2019.
- C. Bian, C. Feng, C. Qian and Y. Yu. An efficient evolutionary algorithm for subset selection with general cost constraints. In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, pp.3267-3274, New York, NY, 2020.

References

- C. Qian, Y. Yu and K. Tang. Approximation guarantees of stochastic greedy algorithms for subset selection. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18),* pp. 1478–1484, Stockholm, Sweden, 2018.
- C. Qian, Y. Yu, K. Tang, X. Yao and Z.-H. Zhou. Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artificial Intelligence,* 2019, 275: 279-294.
- C. Harshaw, M. Feldman, J. Ward and A. Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In: *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, pp. 2634–2643, Long Beach, CA, 2019.
- C. Bian, C. Feng, C. Qian and Y. Yu. An efficient evolutionary algorithm for subset selection with general cost constraints. In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, pp.3267-3274, New York, NY, 2020.

References

- Roostapour, V., Neumann, A., Neumann, F., and Friedrich, T. 2019. Pareto optimization for subset selection with dynamic cost constraints. The Thirty-Third AAAI Conference on Artificial Intelligence 2019, pp. 2354-2361. AAAI Press.
- Albert, R., and Barabási, A.-L. 2002. Statistical mechanics of complex networks. Reviews of modern physics, pp. 74(1):47.
- Barbieri, N., Bonchi, F., and Manco, G. 2012. Topic-aware social influence propagation models. In IEEE Conference on Data Mining, pp. 81–90. IEEE Computer Society.
- Hogg, T., and Lerman, K. 2012. Social dynamics of Digg. EPJ Data Science 1(1):5.
- Kempe, D., Kleinberg, J. M., and Tardos, E. 2015. Maximizing the spread of influence through a social network. Theory of Computing, pp. 11:105–147.
- Krause, A., and Golovin, D. 2014. Submodular function maximization. In Bordeaux, L.; Hamadi, Y.; and Kohli, P., eds., Tractability: Practical Approaches to Hard Problems. Cambridge University Press, pp. 71–104.

References

- Lee, J., Sviridenko, M., and Vondra'k, J. 2010. Submodular maximization over multiple matroids via generalized exchange properties. Math. Oper. Res. 35(4):795–806.
- Friedrich, T., and Neumann, F. 2015. Maximizing submodular functions under matroid constraints by evolutionary algorithms. Evolutionary Computation 23(4):543–558.
- Doerr, B., Doerr, C., Neumann, A., Neumann, F., Sutton, A.M. 2020. Optimization of chance-constrained submodular functions. The Thirty-Fourth AAAI Conference on Artificial Intelligence 2020, pp. 1460–1467. AAAI Press.
- Chebyshev, P. 1867. Des valeurs moyennes. Liouville's J Math Pure Appl 12:177–184.
- Chernoff, H. 1952. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Ann. Math. Statist. 23(4):493–507.
- Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., Van- Briesen, J. M., and Glance, N. S. 2007. Cost-effective outbreak detection in networks. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2007, pp. 420–429. ACM.

References

- Kempe, D., Kleinberg, J. M., and Tardos, E. 2003. Maximizing the spread of influence through a social network. The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2003, pp. 137–146. ACM.
- Neumann, A. and Neumann, F. 2020. Optimising monotone chance-constrained submodular functions using evolutionary multi-objective algorithms. Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference, PPSN 2020. Lecture Notes in Computer Science, pp. 404-417. Springer.
- Motwani, R., Raghavan, P. 1995. Randomized algorithms. Cambridge University Press.
- Doerr, B., Neumann, F. 2020. Theory of Evolutionary Computation – Recent developments in discrete optimization. Natural Computing Series, Springer.
- Assimi, H., Harper, O., Xie, Y., Neumann, A., Neumann, F. 2020. Evolutionary bi-objective optimization for the dynamic chance-constrained knapsack problem based on tail bound objectives. The 24th European Conference on Artificial Intelligence, ECAI 2020, pp. 307—314. IOS Press.
- Xie, Y., Harper, O., Assimi, H., Neumann, A., Neumann, F. 2019. Evolutionary algorithms for the chance-constrained knapsack problem. The Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 338–346. ACM.

References

- Xie, Y., Neumann, A., Neumann, F. 2020. Specific single- and multi-objective evolutionary algorithms for the chance-constrained knapsack problem. The Genetic and Evolutionary Computation Conference, GECCO 2020, pp. 271–279, ACM.
- Giel, O., Wegener, I. 2003. Evolutionary algorithms and the maximum matching problem. In: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science, STACS 2003. Lecture Notes in Computer Science, vol. 2607, pp. 415–426. Springer.
- Doerr, B., Doerr, C., Neumann, A., Neumann, F., Sutton, A. M. 2020. Optimization of chance-constrained submodular functions. The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, pp. 1460–1467. AAAI Press.
- Feige, U. 1998. A threshold of ln n for approximating set cover. J. ACM 45(4), pp. 634–652.
- Khuller, S., Moss, A., Naor, J. 1999. The budgeted maximum coverage problem. Information Processing Letters 70(1), pp. 39–45.