10 Approximation Algorithms

8 July 2025

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg
version `2025-07-08 10:12`

Outline

10 Approximation Algorithms

10.1 Motivation and Definitions

Recap: Optimization Problems, NPO

Recall general optimization problem $U \in \mathsf{NPO}$:

- each instance x has non-empty set of *feasible solutions* $M(x)$
- objective function *cost* assigns value $cost(y)$ to all candidate solutions $y \in M(x)$
- can check in polytime
  - whether x is a valid instance
  - whether $y \in M(x)$
  - compute $cost(y) \in \mathbb{Q}$

Recap: Optimization Problems, NPO

Recall general optimization problem $U \in \mathsf{NPO}$:

- each instance x has non-empty set of *feasible solutions* $M(x)$
- objective function *cost* assigns value $cost(y)$ to all candidate solutions $y \in M(x)$
- can check in polytime
  - whether x is a valid instance
  - whether $y \in M(x)$
  - compute $cost(y) \in \mathbb{Q}$

For each U, consider two variants:

- *optimization problem:* output $y \in M(x)$ s.t. $cost(y) = \overset{\text{min or max}}{goal}_{y' \in M(x)} cost(y')$
- *evaluation problem:* output $goal_{y \in M(x)} cost(y)$

Perfect is the enemy of good

Optimal solutions are great, but if they are too expensive to get,
maybe ***"close-to-optimal"*** suffices?

A "consistent" with problem

A ***heuristic*** is an algorithm A that always computes a feasible solution $A(x) \in M(x)$,
but we may not have any guarantees about $cost(A(x))$.

(Sometimes that's all we have . . .)

Perfect is the enemy of good

Optimal solutions are great, but if they are too expensive to get,
maybe ***"close-to-optimal"*** suffices?

A "consistent" with problem

A ***heuristic*** is an algorithm A that always computes a feasible solution $A(x) \in M(x)$,
but we may not have any guarantees about $cost(A(x))$.

(Sometimes that's all we have . . .)

Our goal: Prove guarantees about worst possible $cost(A(x))$.
Problem: optimal objective function value depends on x,
so how to define *"good enough"*?

Perfect is the enemy of good

Optimal solutions are great, but if they are too expensive to get,
maybe ***"close-to-optimal"*** suffices?

A "consistent" with problem

A ***heuristic*** is an algorithm A that always computes a feasible solution $A(x) \in M(x)$,
but we may not have any guarantees about $cost(A(x))$.

(Sometimes that's all we have . . .)

Our goal: Prove guarantees about worst possible $cost(A(x))$.
Problem: optimal objective function value depends on x,
so how to define *"good enough"*?

Relate $cost(A(x))$ to ***OPT*** $= goal_{y \in M(x)} cost(y)$. $\rightsquigarrow$ *approximation algorithm*

Approximation Algorithms

Definition 10.1 (Approximation Ratio)

Let $U = (\Sigma_I, \Sigma_O, L, L_I, M, cost, goal)$ be an optimization problem. For every $x \in L_I$ we denote its *optimal objective value* by $OPT = OPT_U(x) = goal_{y \in M(x)} cost(y)$.

Let further A be an algorithm consistent with U. $A(x) \in M(x)$

The ***approximation ratio*** $R_A(x)$ ***of*** A ***on*** x is defined as $R_A(x) = \dfrac{cost(A(x))}{OPT_U(x)}$. ◀

Note: For minimization problems, $R_A \geq 1$; for maximization problems $R_A \leq 1$

Approximation Algorithms

Definition 10.1 (Approximation Ratio)

Let $U = (\Sigma_I, \Sigma_O, L, L_I, M, cost, goal)$ be an optimization problem. For every $x \in L_I$ we denote its *optimal objective value* by $OPT = OPT_U(x) = goal_{y \in M(x)} cost(y)$.

Let further A be an algorithm consistent with U.

The ***approximation ratio $R_A(x)$ of A on x*** is defined as $R_A(x) = \dfrac{cost(A(x))}{OPT_U(x)}$.

Note: For minimization problems, $R_A \geq 1$; for maximization problems $R_A \leq 1$

Definition 10.2 (Approximation Algorithm)

An algorithm A consistent with an optimization problem $U = (\Sigma_I, \Sigma_O, L, L_I, M, cost, goal)$ is called a ***c-approximation (algorithm) for U*** if

- $goal = \min$ and $\forall x \in L_I : R_A(x) \leq c$;
- $goal = \max$ and $\forall x \in L_I : R_A(x) \geq c$.

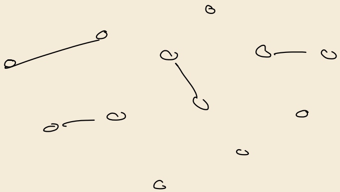
10.2 Vertex Cover and Matchings

Example: Vertex Cover

Recall the VertexCover optimization problem.
C is a VC iff $\{u, v\} \in E : \{u, v\} \cap C \neq \emptyset$
$goal = \min$
How can we vouch for a VC C to be (close to) optimal?

Need a way to lower bound OPT!

Example: Vertex Cover



Recall the VertexCover optimization problem.

C is a VC iff $\{u, v\} \in E : \{u, v\} \cap C \neq \emptyset$

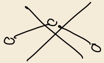
goal = min

How can we vouch for a VC C to be (close to) optimal?

Definition 10.3 ((Maximal/Maximum/Perfect) Matching)

Given graph $G = (V, E)$, a set $M \subseteq E$ is a *matching* (in G) if (V, M) has max-degree 1.

disjoint pairs of vertices



M is *(⊆-) maximal* (a.k.a. *saturated*) if no superset of M is a matching.

M is a *maximum matching* is there is no matching of strictly larger cardinality in G.

M is a *perfect matching* if $|M| = |V|/2$.

Example: Vertex Cover

Recall the VertexCover optimization problem.
C is a VC iff $\{u,v\} \in E : \{u,v\} \cap C \neq \emptyset$
$goal = \min$
How can we vouch for a VC C to be (close to) optimal?

Definition 10.3 ((Maximal/Maximum/Perfect) Matching)

Given graph $G = (V, E)$, a set $M \subseteq E$ is a *matching* (in G) if (V, M) has max-degree 1.

disjoint pairs of vertices

M is *(⊆-) maxima**l*** (a.k.a. *saturated*) if no superset of M is a matching.
M is a *maxim**um** matching* is there is no matching of strictly larger cardinality in G.
M is a *perfect matching* if $|M| = |V|/2$.

Note:

- $\subseteq$-maximal matchings easy to find via greedy algorithm.
- Maxim**um** matchings are much more complicated, but also computable in polytime (Edmonds's "Blossom algorithm")

Matching → Vertex Cover

Lemma 10.4 (VC ≥ M)

If M is a matching and C is a vertex cover in G, then $|C| \geq |M|$.

Matching → Vertex Cover

Lemma 10.4 (VC ≥ M)

If M is a matching and C is a vertex cover in G, then $|C| \geq |M|$.

Proof:

Let $\{v, w\} \in M \subseteq E$. $\rightsquigarrow$ C has to contain v or w (or both).

```
procedure matchingVertexCoverApprox(G = (V, E))
    // greedy maximal matching
    M := ∅
    for e ∈ E // arbitrary order
        if M ∪ {e} is a matching
            M := M ∪ {e}
    return ⋃_{{u,v}∈M} {u, v}
```

Theorem 10.5 (Matching is 2-approx for Vertex Cover)

matchingVertexCoverApprox is a *2-approximation* for VertexCover.

Matching → Vertex Cover

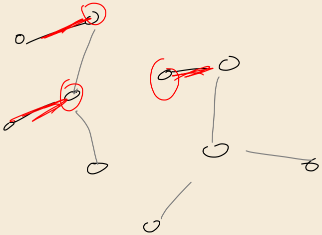
Lemma 10.4 (VC ≥ M)

If M is a matching and C is a vertex cover in G, then $|C| \geq |M|$.

Proof:

Let $\{v, w\} \in M \subseteq E$. $\rightsquigarrow$ C has to contain v or w (or both).

Since all $|M|$ matching edges are disjoint, C must cover them by $\geq |M|$ distinct endpoint. ■



Matching → Vertex Cover

Lemma 10.4 (VC ≥ M)

If M is a matching and C is a vertex cover in G, then $|C| \geq |M|$.

Proof:

Let $\{v, w\} \in M \subseteq E$. $\rightsquigarrow$ C has to contain v or w (or both).

Since all $|M|$ matching edges are disjoint, C must cover them by $\geq |M|$ distinct endpoint. $\blacksquare$

```
procedure matchingVertexCoverApprox(G = (V, E))
    // greedy maximal matching
    M := ∅
    for e ∈ E // arbitrary order
        if M ∪ {e} is a matching
            M := M ∪ {e}
    return ⋃_{{u,v}∈M} {u, v}
```

Matching → Vertex Cover

Lemma 10.4 (VC ≥ M)

If M is a matching and C is a vertex cover in G, then $|C| \geq |M|$.

Proof:

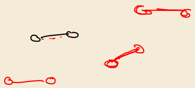
Let $\{v, w\} \in M \subseteq E$. $\rightsquigarrow$ C has to contain v or w (or both).

Since all $|M|$ matching edges are disjoint, C must cover them by $\geq |M|$ distinct endpoint. ■

```
procedure matchingVertexCoverApprox(G = (V, E))
    // greedy maximal matching
    M := ∅
    for e ∈ E // arbitrary order
        if M ∪ {e} is a matching
            M := M ∪ {e}
    return ⋃_{{u,v}∈M} {u, v}  = C
```

(1) C is VC

otherwise some edge uncovered



(2) 2-approx

$|C| = 2|M|$

$OPT \geq |M|$

$\frac{|C|}{OPT} \leq \frac{2|M|}{|M|} = 2$

Theorem 10.5 (Matching is 2-approx for Vertex Cover)

matchingVertexCoverApprox is a *2-approximation* for VertexCover.

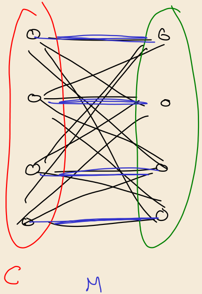
Can we do better?

Maybe do smarter analysis?

Can we do better?

Maybe do smarter analysis?

A tight example for "VC $\geq$ M": $K_{n,n}$



Can we do better?

Maybe do smarter analysis?

A tight example for "VC $\geq$ M": $K_{n,n}$

$\varepsilon > 0$ constant

Assuming the *unique games conjecture*, no polytime $(2 - \varepsilon)$ approx for VC.

Simple matching-based approximation worst-case optimal . . .

10.3 The Drosophila of Approximation: Set Cover

(Weighted) Set Cover

Definition 10.6 (SetCover)

Given: a number n, $\mathcal{S} = \{S_1, \ldots, S_k\}$ of k subsets of $U = [n]$,
and a cost function $c : S \to \mathbb{N}$.

Solutions: $\mathcal{C} \subseteq [k]$ with $\bigcup_{i \in \mathcal{C}} S_i = U$

Cost: $\sum_{i \in \mathcal{C}} c(S_i)$

Goal: min

- *cardinality version* a.k.a. UnweightedSetCover has cost $c(S) = |S|$
- UnweightedSetCover generalizes VertexCover:
  For VertexCover instances, the sets S_i are the sets of edges incident at a vertex v
  $\rightsquigarrow$ additional property that each $e \in U$ occurs in **exactly** 2 sets S_i
- general UnweightedSetCover = Vertex Cover on hypergraphs

(Weighted) Set Cover

Definition 10.6 (SetCover)

Given: a number n, $\mathcal{S} = \{S_1, \ldots, S_k\}$ of k subsets of $U = [n]$, and a cost function $c : S \to \mathbb{N}$.

Solutions: $\mathcal{C} \subseteq [k]$ with $\bigcup_{i \in \mathcal{C}} S_i = U$

Cost: $\sum_{i \in \mathcal{C}} c(S_i)$

Goal: min

- *cardinality version* a.k.a. UnweightedSetCover has cost $c(S) = |S|$
- UnweightedSetCover generalizes VertexCover:
  For VertexCover instances, the sets S_i are the sets of edges incident at a vertex v
  $\rightsquigarrow$ additional property that each $e \in U$ occurs in **exactly** 2 sets S_i
- general UnweightedSetCover = Vertex Cover on hypergraphs

We will use SetCover to illustrate various techniques for approximation algorithms.