Two-sided Online Bipartite Matching and Vertex Cover: Beating the Greedy Algorithm

by Yajun Wang and Sam Chiu-wai Wong

presented by Maciej Sanocki

16.11.2023

Table of Contents

Table of Contents

Ski rental problem

We know that renting skis costs a and buying them costs b. If we haven't bought skis yet, each time we are going skiing we can either decide to rent or to buy skis. Our goal is to minimize cost.

Ski rental problem

We know that renting skis costs a and buying them costs b. If we haven't bought skis yet, each time we are going skiing we can either decide to rent or to buy skis. Our goal is to minimize cost.

competitiveness

optimal algorithm for this problem is $\frac{1}{(1-1/e)}$-competitive

Online Bipartite b-Matching problem (**OBM**)

Initially, we are given L and their capacity $b_u, u \in L$. At each step a vertex $v \in R$ is revealed with b_v and its incident edges. An algorithm for *OBM* maintains a *b-matching* x, and must irrevocably decide at each step on the values of x_e for each new edge e. The objective is to *maximize* the size of x.

$$\max \sum_{e \in E} x_e$$

$$s.t. \forall_{v \in (L \cup R) \cap T} \sum_{u \in N(v) \cap T} x_{uv} \leq b_v$$

b_5 5

b_4 4

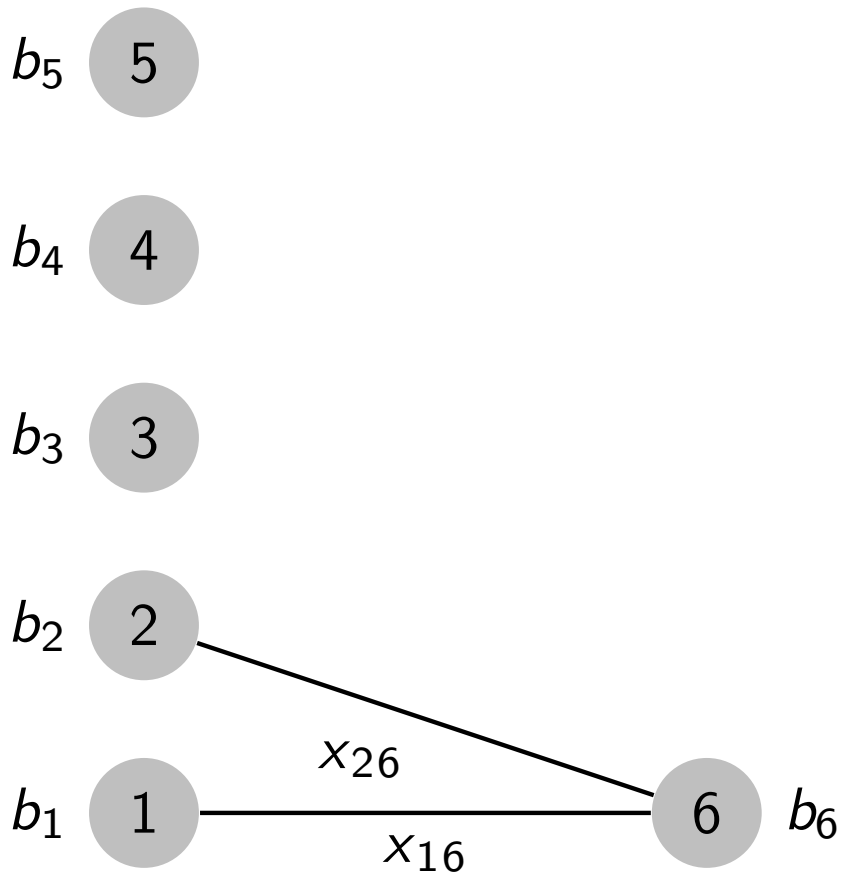
b_3 3

b_2 2

b_1 1

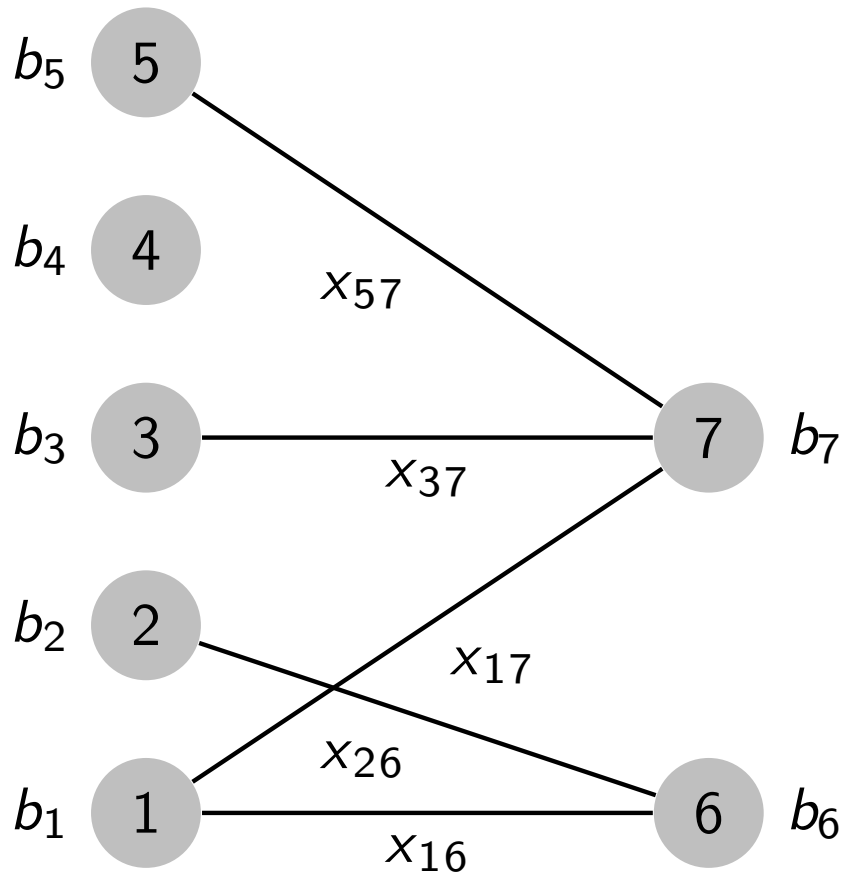
$$\max \sum_{e \in E} x_e$$

$$s.t. \forall_{v \in (L \cup R) \cap T} \sum_{u \in N(v) \cap T} x_{uv} \leq b_v$$



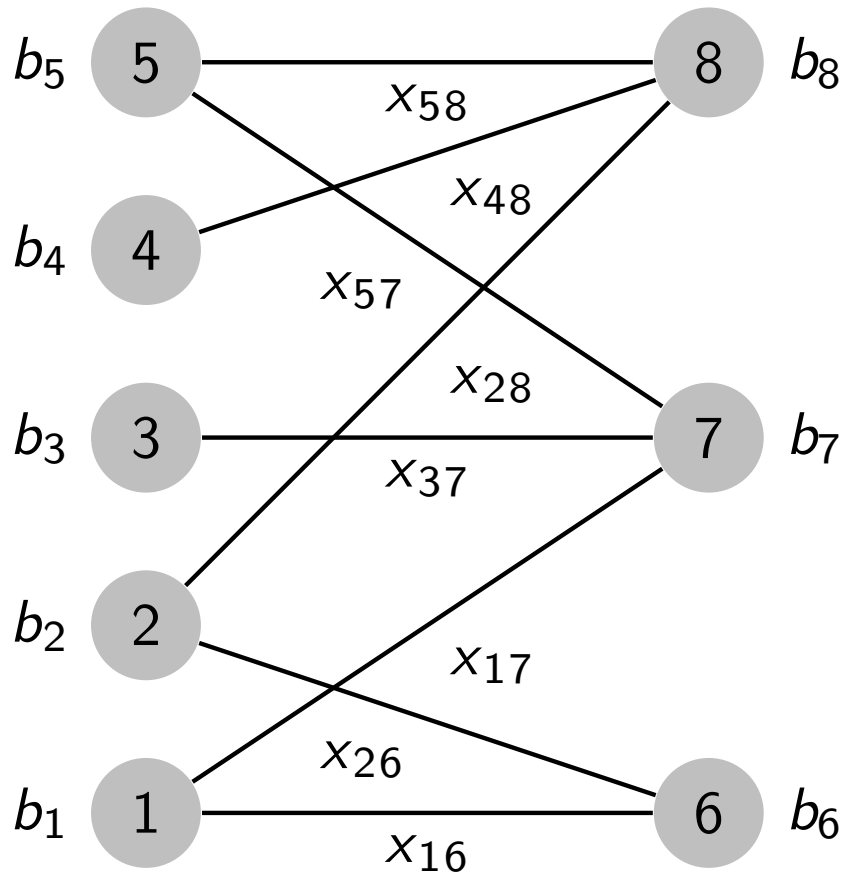
$$\max \sum_{e \in E} x_e$$

$$s.t. \forall_{v \in (L \cup R) \cap T} \sum_{u \in N(v) \cap T} x_{uv} \leq b_v$$



$$\max \sum_{e \in E} x_e$$

$$s.t. \forall_{v \in (L \cup R) \cap T} \sum_{u \in N(v) \cap T} x_{uv} \leq b_v$$



Online Bipartite b-Matching problem (**OBM**)

Initially, we are given L and their capacity $b_u, u \in L$. At each step a vertex $v \in R$ is revealed with b_v and its incident edges. An algorithm for *OBM* maintains a *b-matching* x, and must irrevocably decide at each step on the values of x_e for each new edge e. The objective is to *maximize* the size of x.

Online Bipartite b-Matching problem (**OBM**)

Initially, we are given L and their capacity $b_u, u \in L$. At each step a vertex $v \in R$ is revealed with b_v and its incident edges. An algorithm for *OBM* maintains a *b-matching* x, and must irrevocably decide at each step on the values of x_e for each new edge e. The objective is to *maximize* the size of x.

Greedy algorithm is $1/2$-competitive.

Online Bipartite b-Matching problem (**OBM**)

Initially, we are given L and their capacity $b_u, u \in L$. At each step a vertex $v \in R$ is revealed with b_v and its incident edges. An algorithm for *OBM* maintains a *b-matching* x, and must irrevocably decide at each step on the values of x_e for each new edge e. The objective is to *maximize* the size of x.

Greedy algorithm is $1/2$-competitive.

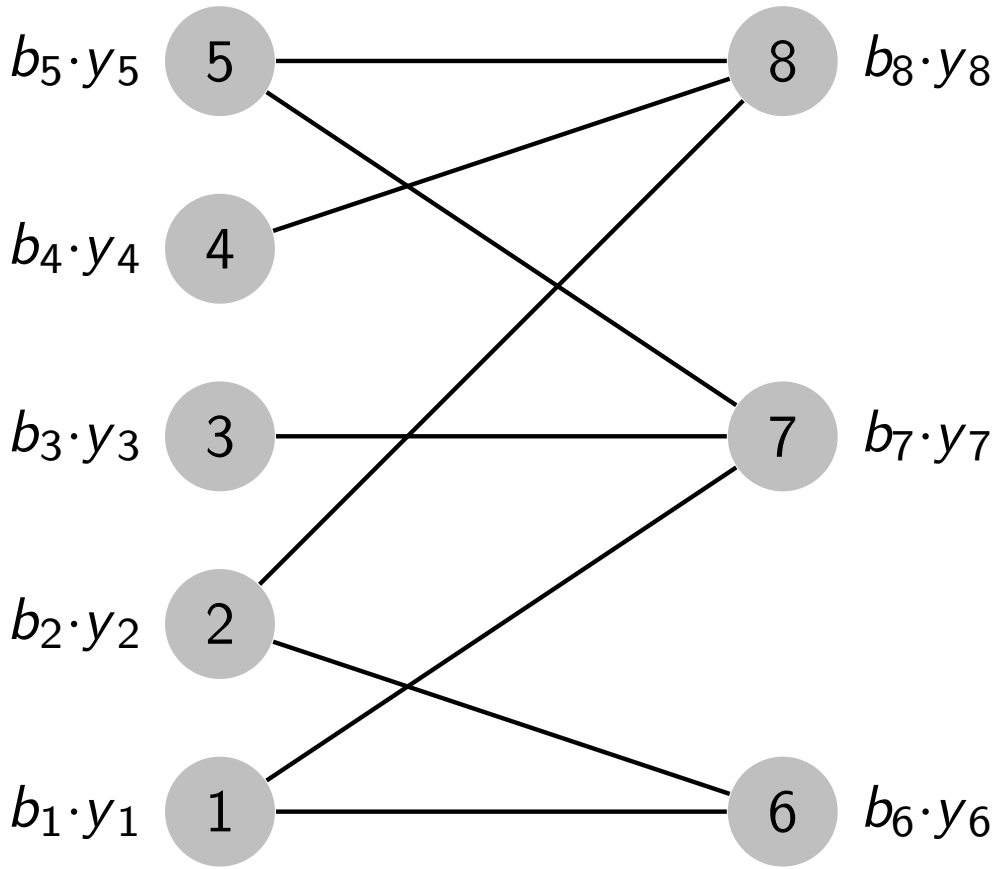
Optimal algorithm for this problem is $(1 - 1/e)$-competitive

Online Bipartite Vertex Cover problem (**OBVC**)

Similarly, for *OBVC* we are required to maintain a vertex cover C at all time by only inserting vertices into C, i.e. no vertex removal is allowed. Thus at each step the task is essentially deciding if v or $N(v)$ should be inserted to C. The objective is to minimize the *weight* of C, $b(C) := \sum_{v \in C} b_v$

$$\min \sum_{v \in V} b_v \cdot y_v$$

$$s.t. y_u + y_v \geq 1, \forall (u, v) \in (E \cap T^2) :$$



Online Bipartite Vertex Cover problem (**OBVC**)

Similarly, for *OBVC* we are required to maintain a vertex cover C at all time by only inserting vertices into C, i.e. no vertex removal is allowed. Thus at each step the task is essentially deciding if v or $N(v)$ should be inserted to C. The objective is to minimize the *weight* of C, $b(C) := \sum_{v \in C} b_v$

Online Bipartite Vertex Cover problem (**OBVC**)

Similarly, for *OBVC* we are required to maintain a vertex cover C at all time by only inserting vertices into C, i.e. no vertex removal is allowed. Thus at each step the task is essentially deciding if v or $N(v)$ should be inserted to C. The objective is to minimize the *weight* of C, $b(C) := \sum_{v \in C} b_v$

Greedy algorithm is 2/1-competitive.

Online Bipartite Vertex Cover problem (**OBVC**)

Similarly, for *OBVC* we are required to maintain a vertex cover C at all time by only inserting vertices into C, i.e. no vertex removal is allowed. Thus at each step the task is essentially deciding if v or $N(v)$ should be inserted to C. The objective is to minimize the *weight* of C, $b(C) := \sum_{v \in C} b_v$

Greedy algorithm is 2/1-competitive.

Can we do better?

Online Bipartite Vertex Cover problem (**OBVC**)

Similarly, for *OBVC* we are required to maintain a vertex cover C at all time by only inserting vertices into C, i.e. no vertex removal is allowed. Thus at each step the task is essentially deciding if v or $N(v)$ should be inserted to C. The objective is to minimize the *weight* of C, $b(C) := \sum_{v \in C} b_v$

Greedy algorithm is $2/1$-competitive.

Can we do better?

We know, that it is at most $(1 - 1/e)$, since it is a generalization of ski rental problem.

Two-sided Online Bipartite b-Matching and Vertex Cover (**TOBM & TOBVC**)

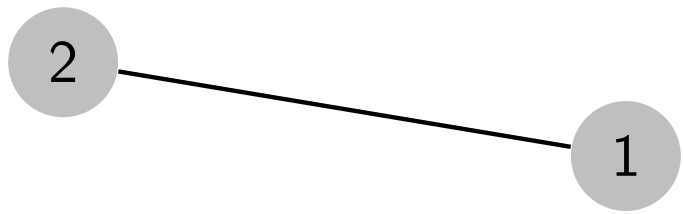
Two-sided version relaxes the constraint that only the right vertices are online.

1

Greedy algorithms for both problems are 1/2-competitive.

Two-sided Online Bipartite b-Matching and Vertex Cover (**TOBM & TOBVC**)

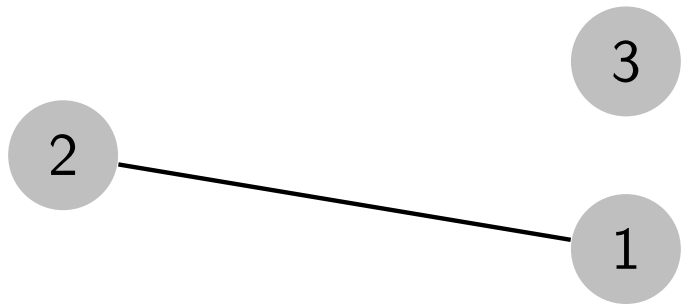
Two-sided version relaxes the constraint that only the right vertices are online.



Greedy algorithms for both problems are 1/2-competitive.

Two-sided Online Bipartite b-Matching and Vertex Cover (**TOBM & TOBVC**)

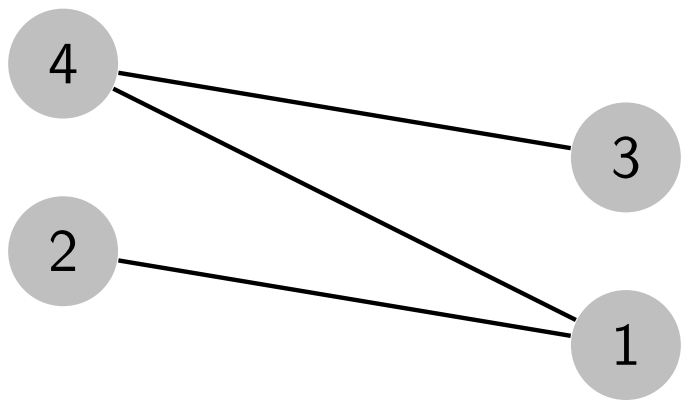
Two-sided version relaxes the constraint that only the right vertices are online.



Greedy algorithms for both problems are 1/2-competitive.

Two-sided Online Bipartite b-Matching and Vertex Cover (**TOBM & TOBVC**)

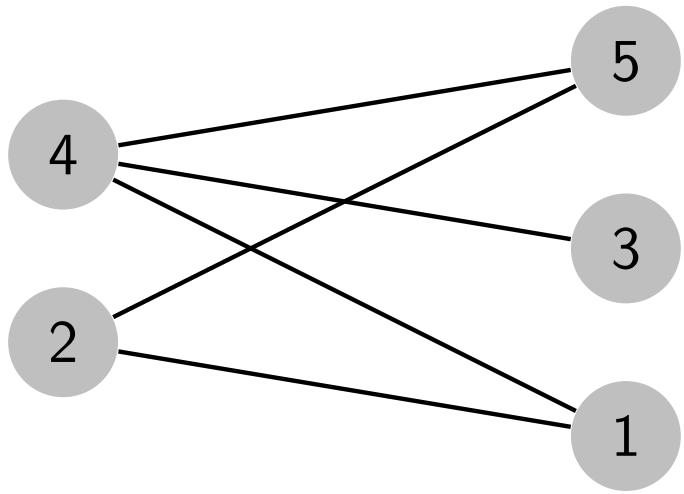
Two-sided version relaxes the constraint that only the right vertices are online.



Greedy algorithms for both problems are 1/2-competitive.

Two-sided Online Bipartite b-Matching and Vertex Cover (**TOBM & TOBVC**)

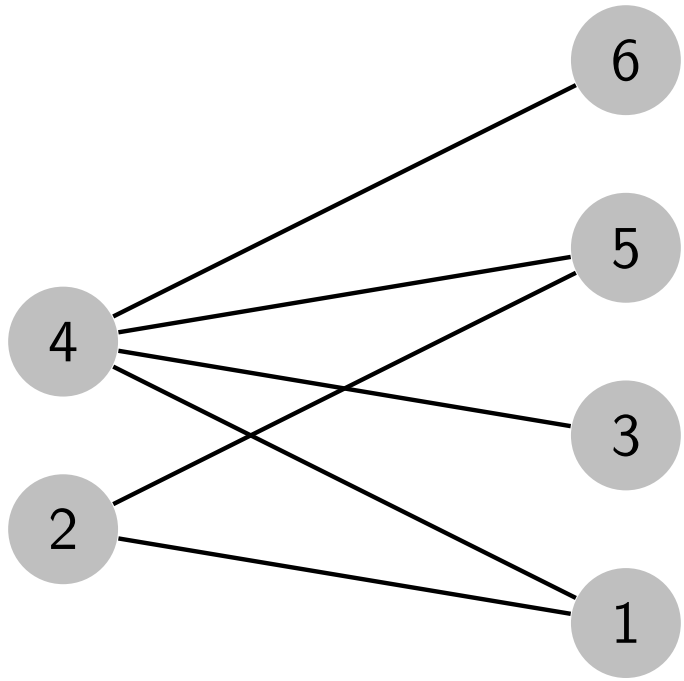
Two-sided version relaxes the constraint that only the right vertices are online.



Greedy algorithms for both problems are 1/2-competitive.

Two-sided Online Bipartite b-Matching and Vertex Cover (**TOBM & TOBVC**)

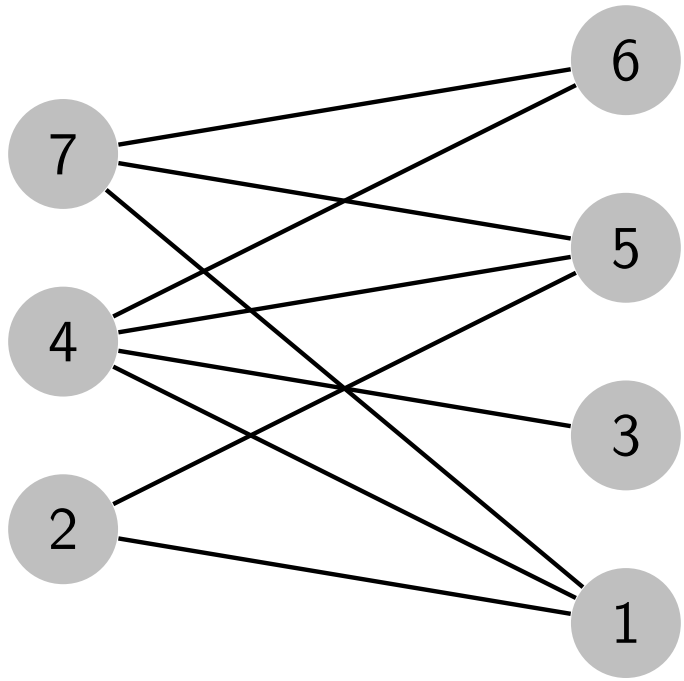
Two-sided version relaxes the constraint that only the right vertices are online.



Greedy algorithms for both problems are 1/2-competitive.

Two-sided Online Bipartite b-Matching and Vertex Cover (**TOBM & TOBVC**)

Two-sided version relaxes the constraint that only the right vertices are online.



Greedy algorithms for both problems are 1/2-competitive.

Fractional vertex cover and b-matching

When analysing *fractional* versions, we enable y's and x's to be fractions rather then natural numbers.

Table of Contents

Connection between Bipartite Vertex Cover and Bipartite b-Matching

$$\max \sum_{e \in E} x_e$$

$$s.t. \sum_{u \in N(v)} x_{uv} \leq b_v, \forall v \in V$$

$$x \geq 0$$

$$\min \sum_{v \in V} b_v \cdot y_v$$

$$s.t. y_u + y_v \geq 1, \forall (u, v) \in E$$

$$y \geq 0$$

Connection between Bipartite Vertex Cover and Bipartite b-Matching

$$\max \sum_{e \in E} x_e$$

$$s.t. \sum_{u \in N(v)} x_{uv} \leq b_v, \forall v \in V$$

$$x \geq 0$$

$$\min \sum_{v \in V} b_v \cdot y_v$$

$$s.t. y_u + y_v \geq 1, \forall (u, v) \in E$$

$$y \geq 0$$

these problems are dual to each other!

Connection between Bipartite Vertex Cover and Bipartite b-Matching

$$\max \sum_{e \in E} x_e$$

$$s.t. \sum_{u \in N(v)} x_{uv} \leq b_v, \forall v \in V$$

$$x \geq 0$$

$$\min \sum_{v \in V} b_v \cdot y_v$$

$$s.t. y_u + y_v \geq 1, \forall (u, v) \in E$$

$$y \geq 0$$

these problems are dual to each other!

$$\max c^T x$$

$$\text{s.t.} Ax \leq b, x \geq 0.$$

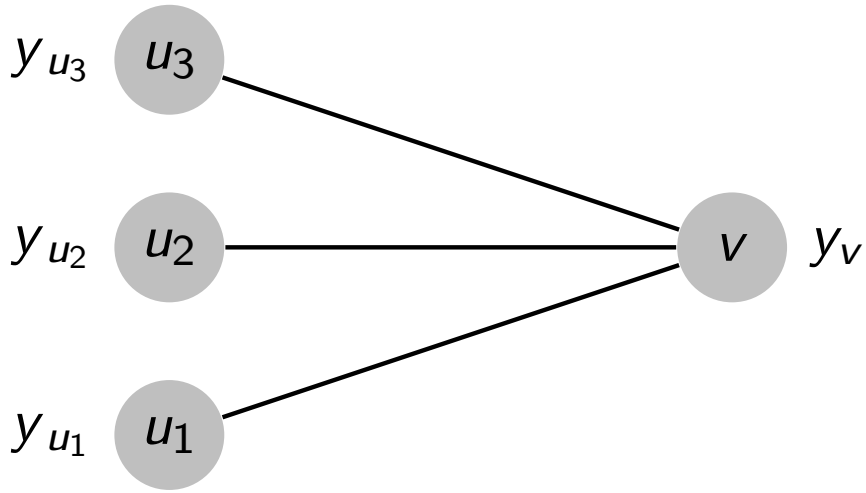
$$\min b^T y$$

$$\text{s.t.} A^T y \geq c, y \geq 0.$$

Lemma 1. *(Lossless rounding) Any deterministic algorithm for fractional* **OBVC** *can be converted to a randomized algorithm for integral* **OBVC** *with the same competitive ratio.*

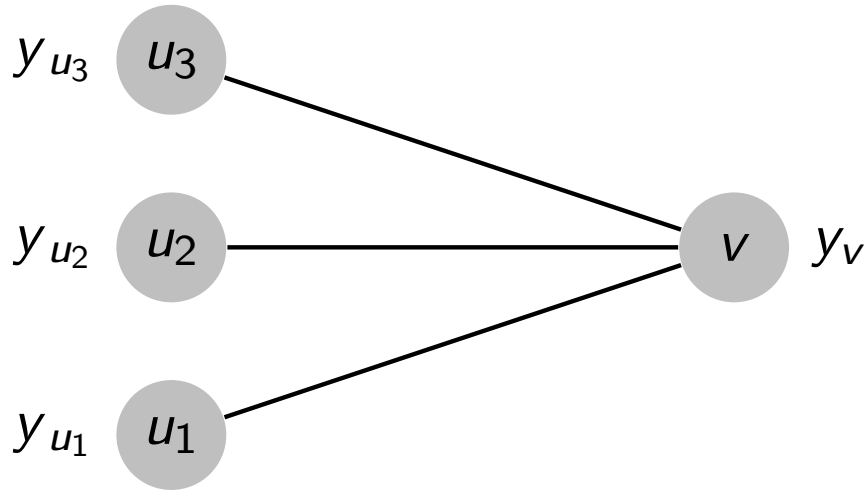
(One-sided) Online Bipartite Vertex Cover

For each vertex v, algorithm should maintain a non-decreasing cover potential y_v.



(One-sided) Online Bipartite Vertex Cover

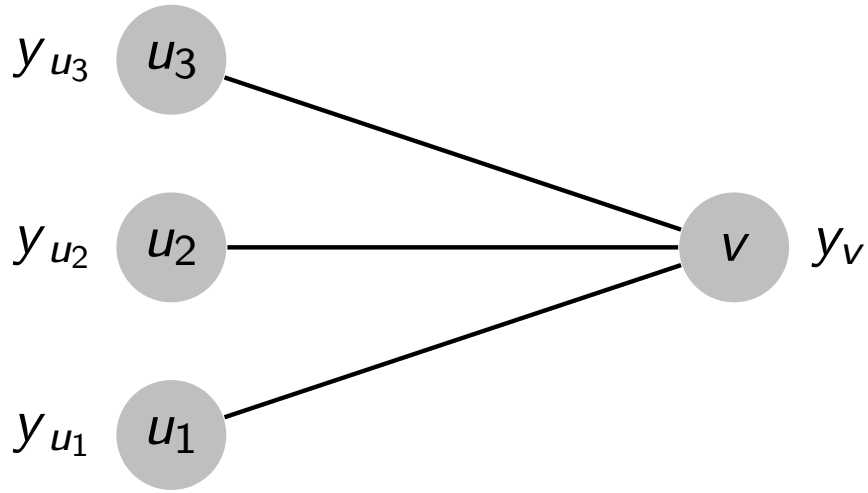
For each vertex v, algorithm should maintain a non-decreasing cover potential y_v.



To maintain a feasible vertex cover: $\forall_{u \in N(V)} y_v + y_u \geq 1$

(One-sided) Online Bipartite Vertex Cover

For each vertex v, algorithm should maintain a non-decreasing cover potential y_v.

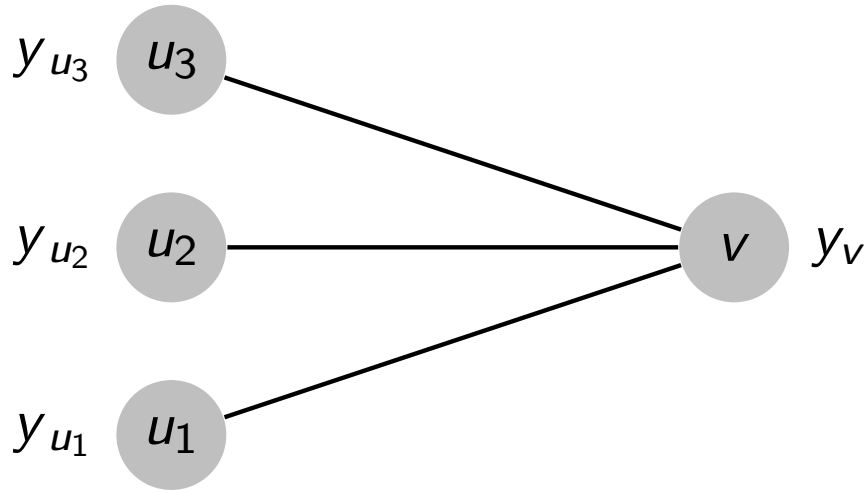


To maintain a feasible vertex cover: $\forall_{u \in N(V)} y_v + y_u \geq 1$

Suppose we set $y_v = 1 - y$ for some y.

(One-sided) Online Bipartite Vertex Cover

For each vertex v, algorithm should maintain a non-decreasing cover potential y_v.



To maintain a feasible vertex cover: $\forall_{u \in N(V)} y_v + y_u \geq 1$

Suppose we set $y_v = 1 - y$ for some y.

To maintain a feasible vertex cover, any $y_u < y$ for $u \in N(v)$ should be increased to y.

The crux lies in how y is determined.

The crux lies in how y is determined.

Total potential increment is equal to:

$$b_v(1-y) + \sum_{u \in N(v): y_u < y} b_u(y - y_u)$$

The crux lies in how y is determined.

Total potential increment is equal to:

$$b_v(1-y) + \sum_{u \in N(v): y_u < y} b_u(y - y_u)$$

Choosing total potential increment to be at most $\frac{b_v}{1-1/e}$ seems to be a reasonable option.

The water-filling algorithm for **OBVC**

Algorithm 1 Water-filling algorithm for OBVC

Input. L *and* b_u, $u \in L$

Initialize for each $u \in L, y_u = 0$;

for *each online vertex* $v \in R$ **do**

  Maximize $y \leq 1$,

    $s.t.\ b_v(1-y) + \sum_{u \in N(v)} b_u \max\{y - y_u, 0\} \leq b_v/(1-1/e)$;

  For each $u \in N(v)$, $y_u \leftarrow \max\{y_u, y\}$;

  $y_u \leftarrow 1 - y$;

end

Short analysis

Let C^* be a minimum vertex cover of G.

Short analysis

Let C^* be a minimum vertex cover of G.

Our strategy is to charge the potential increment to vertices of C^* in such a way that each vertex of $v \in C^*$ is charged at most $\frac{b_v}{1-1/e}$.

Let v be the current online vertex. Our algorithm sets $y_v = 1 - y$ for some y. Let y_u be the potential of $u \in N(v)$. We consider two cases.

Short analysis

Let C^* be a minimum vertex cover of G.

Our strategy is to charge the potential increment to vertices of C^* in such a way that each vertex of $v \in C^*$ is charged at most $\frac{b_v}{1-1/e}$.

Let v be the current online vertex. Our algorithm sets $y_v = 1 - y$ for some y. Let y_u be the potential of $u \in N(v)$. We consider two cases.

Case 1. $v \in C^*$: trivial

Short analysis

Let C^* be a minimum vertex cover of G.

Our strategy is to charge the potential increment to vertices of C^* in such a way that each vertex of $v \in C^*$ is charged at most $\frac{b_v}{1-1/e}$.

Let v be the current online vertex. Our algorithm sets $y_v = 1 - y$ for some y. Let y_u be the potential of $u \in N(v)$. We consider two cases.

Case 1. $v \in C^*$: trivial

Case 2. $v \notin C^*$: We must have $N(v) \subseteq C^*$.

Short analysis

Let C^* be a minimum vertex cover of G.

Our strategy is to charge the potential increment to vertices of C^* in such a way that each vertex of $v \in C^*$ is charged at most $\frac{b_v}{1-1/e}$.

Let v be the current online vertex. Our algorithm sets $y_v = 1 - y$ for some y. Let y_u be the potential of $u \in N(v)$. We consider two cases.

Case 1. $v \in C^*$: trivial

Case 2. $v \notin C^*$: We must have $N(v) \subseteq C^*$. Also we must find a way to charge $b_v(1 - y)$ to $N(v)$

Let $g(y) = \frac{1}{e-1} + y$. Rewrite
$b_v(1-y) + \sum_{u \in N(v)} b_u \max\{y - y_u, 0\} \leq \frac{b_v}{1-1/e}$ as:

$$\sum_{u \in N(v)} b_u \max\{y - y_u, 0\} \leq b_v(\frac{1}{e-1} + y) = b_v g(v)$$

If equality holds, we want to additionally charge
$\frac{1-y}{g(y)} b_u \max\{y - y_u, 0\}$ each $u \in N(v)$ for potential of v.

Since $\frac{1-t}{g(y)}$ is decreasing, $\frac{1-y}{g(y)} b_u(y - y_u)$ can be upper bounded by $b_u \int_{y_u}^{y} \frac{1-t}{g(t)}\, dt$.

Since $\frac{1-t}{g(y)}$ is decreasing, $\frac{1-y}{g(y)} b_u(y - y_u)$ can be upper bounded by $b_u \int_{y_u}^{y} \frac{1-t}{g(t)}\, dt$.

Lemma 2. Let $f : [0,1] \longrightarrow R^+$ be *continuous s.t.* $\frac{1-t}{f(t)}$ is decreasing, and $F(x) = \int_0^x \frac{1-t}{f(t)}\, dt$. If $\sum_{u \in X} b_u(y - y_u) = b_v f(y)$ for some set X and $y > y_u$ for $u \in X$, then

$$b_v(1 - y) \leq \sum_{u \in X} b_u(F(y) - F(y_u))$$

Theorem 1. *Algorithm 1* is $1 - 1/e$-competitive and hence optimal for **OBVC**.

The water-filling algorithm for **TOBVC**

We replace $g(t) = 1/(e-1) + t$ with any continuous function $f : [0,1] \longrightarrow R_+$ for which $\frac{1-t}{f(t)}$ is decreasing.

The water-filling algorithm for **TOBVC**

We replace $g(t) = 1/(e-1) + t$ with any continuous function $f : [0,1] \longrightarrow R_+$ for which $\frac{1-t}{f(t)}$ is decreasing.

Algorithm 3 Water-filling algorithm for TOBVC

Let T be the set of arrived vertices. Initially $T = \emptyset$

for *each online vertex* v **do**

  Maximize $y \leq 1$, *s.t.* $\sum_{u \in N(v) \cap T} b_u \max\{y - y_u, 0\} \leq b_v f(y)$;

  For each $u \in N(v) \cap T$, $y_u \leftarrow \max\{y_u, y\}$;

  $y_v \leftarrow 1 - y$;

  $T \leftarrow T \cup \{v\}$

end

Analysis

What is the sum of charges to each $v \in C^*$

Analysis

What is the sum of charges to each $v \in C^*$

- when v arrives it is charged at most b_v for itself.

Analysis

What is the sum of charges to each $v \in C^*$

- when v arrives it is charged at most b_v for itself.
- v is charged for neighbours who already arrived:$\sum_{u \in N(v)} b_u \max\{y - y_u, 0\}$ which is at most $b_v f(y)$

Analysis

What is the sum of charges to each $v \in C^*$

- when v arrives it is charged at most b_v for itself.
- v is charged for neighbours who already arrived:$\sum_{u \in N(v)} b_u \max\{y - y_u, 0\}$ which is at most $b_v f(y)$
- it may be responsible for neighbours who will arrive in future, by lemma 2. at most: $\int_{1-y}^{1} \frac{1-t}{f(t)}\, dt$

Analysis

What is the sum of charges to each $v \in C^*$

- when v arrives it is charged at most b_v for itself.
- v is charged for neighbours who already arrived:$\sum_{u \in N(v)} b_u \max\{y - y_u, 0\}$ which is at most $b_v f(y)$
- it may be responsible for neighbours who will arrive in future, by lemma 2. at most: $\int_{1-y}^{1} \frac{1-t}{f(t)}\, dt$

Setting $z = 1 - y$, the total charges to each $v \in C^*$ are at most:

$$b_v \beta(f), \text{where } \beta(f) := \max_{z \in [0,1]} 1 + f(1-z) + \int_z^1 \frac{1-t}{f(t)}\, dt$$

Theorem 2. Suppose $f : [0,1] \longrightarrow R_+$ is continuous and $\frac{1-t}{f(t)}$ is decreasing. Let $\beta(f) := \max_{z\in[0,1]} 1 + f(1-z) + \int_z^1 \frac{1-t}{f(t)}\, dt$. Then *Algorithm 2* is $1/\beta(f)$-competitive for **TOBVC**.

Computing the Optimal f

In essence, finding the f(y) to get a small β is to solve the following optimization problem:

$$\inf_{f \in F} \max_{z \in [0,1]} 1 + f(1-z) + \int_z^1 \frac{1-t}{f(t)}\, dt$$

Computing the Optimal f

In essence, finding the f(y) to get a small β is to solve the following optimization problem:

$$\inf_{f \in F} \max_{z \in [0,1]} 1 + f(1-z) + \int_z^1 \frac{1-t}{f(t)}\, dt$$

Theorem 3. Let $f(z) = (\frac{1+k}{2} - z)^{\frac{1+k}{2k}} (z + \frac{k-1}{2})^{\frac{k-1}{2k}}$, where $k \approx 1.997$. *Algorithm 2.* is then 0.526-competitive for **TOBVC**.

Table of Contents

Modifying the Algorithm 2. to compute **TOBM**

let $f(z)$ be the same as in Theorem 3. and $\beta \approx 1.901$

Algorithm 4 Water-filling algorithm for TOBM

Let T be the set of arrived vertices. Initially $T = \emptyset$

for *each online vertex* v **do**

  Maximize $y \leq 1$, *s.t.* $\sum_{u \in N(v) \cap T} b_u \max\{y - y_u, 0\} \leq b_v f(y)$;

  let $X = \{u \in N(v) \cap T | y_u < y\}$

  for *each* $u \in X$ **do**

    $y_u \leftarrow y$;

    $x_{uv} \leftarrow \frac{b_u(y-y_u)}{\beta}(1 + \frac{1-y}{f(y)})$

  end

  For each $u \in (N(v) \cap T) \backslash X$, $x_{uv} \leftarrow 0$;

  $y_v \leftarrow 1 - y$, $T \leftarrow T \cup \{v\}$

end

Invariants

Invariant 1. $b_u \cdot \frac{y_u + f(1-z_u) + \int_{z_u}^{y_u} \frac{1-t}{f(t)}\, dt}{\beta} \geq x_u$

where z_u is the potential of u set upon it's arrival, y_u is current potential of u and $x_u = \sum_{v \in N(u)} x_{uv}$

Invariants

Invariant 1. $b_u \cdot \frac{y_u + f(1-z_u) + \int_{z_u}^{y_u} \frac{1-t}{f(t)}\, dt}{\beta} \geq x_u$

where z_u is the potential of u set upon it's arrival, y_u is current potential of u and $x_u = \sum_{v \in N(u)} x_{uv}$

Invariant 2. $\sum_{u \in T} b_u y_u = \beta \sum_{(u,v) \in E \cap T^2} x_{uv}$

Invariants

Invariant 1. $b_u \cdot \frac{y_u + f(1-z_u) + \int_{z_u}^{y_u} \frac{1-t}{f(t)}\, dt}{\beta} \geq x_u$

where z_u is the potential of u set upon it's arrival, y_u is current potential of u and $x_u = \sum_{v \in N(u)} x_{uv}$

Invariant 2. $\sum_{u \in T} b_u y_u = \beta \sum_{(u,v) \in E \cap T^2} x_{uv}$

Theorem 4. Our algorithm is $1/\beta \approx 0.526$-competitive for **TOBM** and **TOBVC**.

Table of Contents

References

1. Karlin, A.R., Manasse, M.S., Rudolph, L., Sleator, D.D.: Competitive snoopy caching. Algorithmica 3(1), 79–119 (1988)