The maximum cardinality matching problem

Combinatorial optimization

Giovanni Righini

Università degli Studi di Milano

UNIVERSITÀ DEGLI STUDI DI MILANO

Definitions

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a matching is an edge subset $M \subseteq \mathcal{E}$ such that no edge of M is incident to any vertex of $\mathcal{V}$ more than once.

Given a matching M,

- an edge $[i,j] \in \mathcal{E}$ is *matched* $\Leftrightarrow [i,j] \in M$;
- a vertex $i \in \mathcal{V}$ is *matched* $\Leftrightarrow \exists j \in \mathcal{V} : [i,j] \in M$.

Any matching M of $\mathcal{G}$ contains at most $\lfloor n/2 \rfloor$ edges, where $n = |\mathcal{V}|$.

Problem: compute a *matching* of maximum cardinality.

Definitions

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a matching $M \subseteq \mathcal{E}$, an *alternating path* P is a path $[p_1, p_2, \ldots, p_q]$ in $\mathcal{G}$ such that for each consecutive pair of edges $[p_{i-1}, p_i]$ and $[p_i, p_{i+1}]$ along P, one of them is matched and the other is unmatched.

If q is odd, the n. of edges in P is even $\Rightarrow$ P is an *even alternating path*.
If q is even, the n. of edges in P is odd $\Rightarrow$ P is an *odd alternating path*.

An *alternating cycle* is an alternating path $[p_1, p_2, \ldots, p_q]$ in which $p_1 = p_q$.

If an alternating cycle is odd, it contains two adjacent unmatched edges, $[p_1, p_2]$ and $[p_{q-1}, p_q]$.

Definitions

An *augmenting path* in $\mathcal{G}$ w.r.t. M is an odd alternating path where the first and the last vertices are unmatched.

In an augmenting path the first and the last edges, $[p_1, p_2]$ and $[p_{q-1}, p_q]$, are unmatched.

Reversing the matching status of the edges in an augmenting path w.r.t. M produces a new matching M' with $|M'| = |M| + 1$.

Symmetric difference

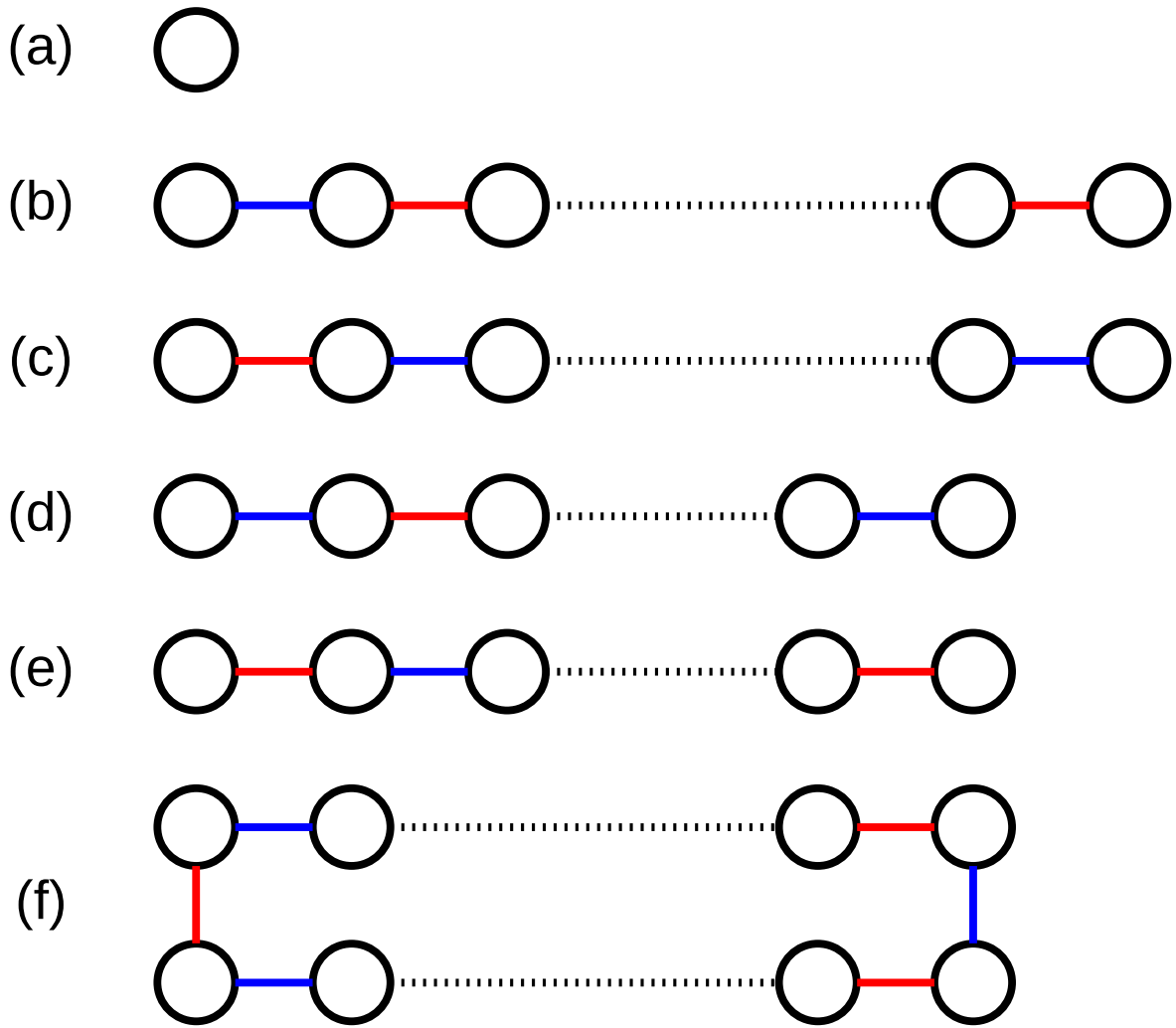
The *symmetric difference* Q of two sets S_1 and S_2 is

$$Q = S_1 \ominus S_2 = (S_1 \cup S_2) - (S_1 \cap S_2).$$

Property 1. Given a matching M and an augmenting path P, the set $M' = M \ominus P$ is a matching and $|M'| = |M| + 1$.
All vertices matched in M are matched in M'; two more vertices are matched in M': the endpoints of P.

Symmetric difference

Property 2. Given M_1 and M_2, $\hat{\mathcal{G}} = (\mathcal{V}, M_1 \ominus M_2)$ has only components of these six types, because $degree(i) \in \{0, 1, 2\}\ \forall i \in \mathcal{V}$.



The augmenting path theorem

Theorem. If a vertex p in unmatched in a matching M and M contains no augmenting path starting at p, then there exists a max cardinality matching in which p is unmatched.

Proof. Let M^* be a max cardinality matching.

If p (unmatched in M) is unmatched in M^*, the statement is true.

If p (unmatched in M) is matched in M^*, then $deg(p) = 1$ in $M \ominus M^*$.

Since p is unmatched in M, then in $M \ominus M^*$ p must be endpoint of an alternating path, starting with an edge of M^* (cases (c) and (e)).

Since p is not the endpoint of an augmenting path, then it cannot be an endpoint of an *odd* alternating path in $M \ominus M^*$ (case (e)).

Then p must be the starting point of an *even* alternating path P in $M \ominus M^*$ (case (c)).

Consider $M' = M^* \ominus P$: M' is a matching; $|M'| = |M^*|$; p is unmatched in M'. Therefore, there exists a maximum cardinality matching M' where p is unmatched. $\square$

Searching for augmenting paths

This theorem is the basis for an algorithm that iteratively looks for augmenting paths and stops when no augmenting path exists from any starting vertex.

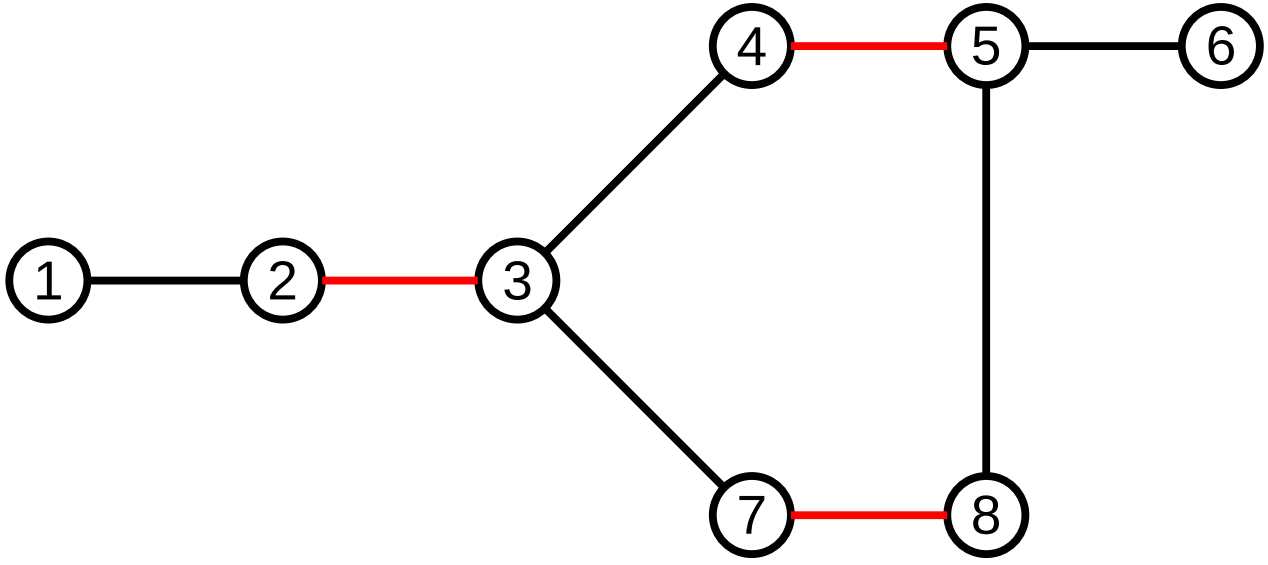
Augmenting paths could be searched by alternately labeling the vertices as “odd” and “even” starting from an unmatched vertex, until another unmatched vertex is reached and it is labeled as “odd”.

From “even” vertices, “odd” vertices can be labeled along unmatched edges.
From “odd” vertices, “even” vertices can be labeled along matched edges.

Such a labeling would work fine in bipartite graphs, but it would not be unique in general graphs because of odd cycles.

Example

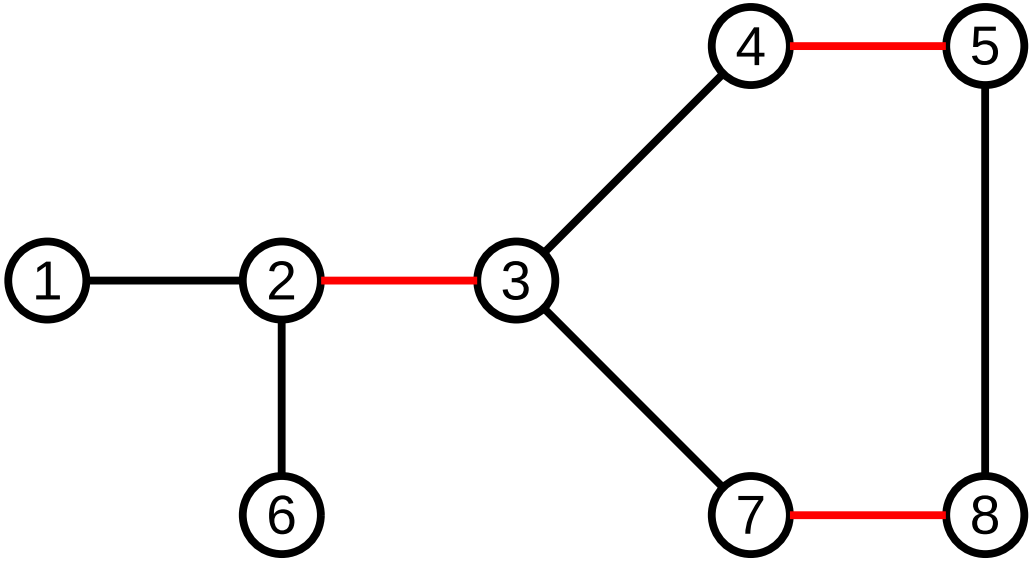
Allowing one label per node would restrict the search too much.



Following $(1, 2, 3, 7, 8, 5)$, $\mathit{label}(5) \leftarrow \mathit{odd}$ and vertex 6 is not reached.
False negative: an augmenting path $(1, 2, 3, 4, 5, 6)$ does exist.

Example

Allowing two labels per node would relax the search too much.

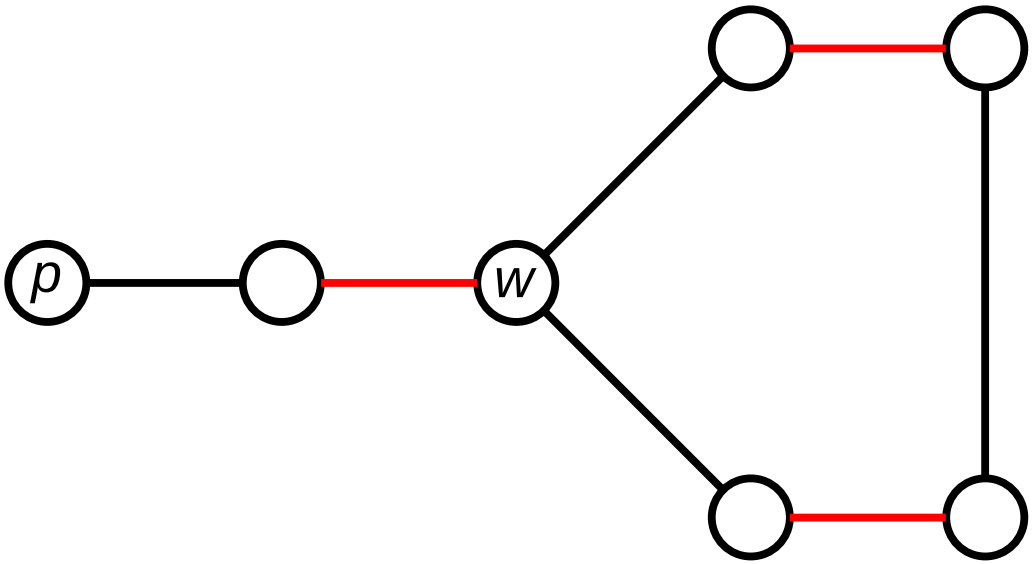


Following $(1, 2, 3, 4, 5, 8, 7, 3, 2, 6)$, a fake augmenting path is found.
False positive: an augmenting path does not exist.

Definitions

Given a graph $\mathcal{G}$ and a matching M, a *flower* is a subgraph made by

- a *stem*: a possibly empty even alternating path from an unmatched vertex p to some vertex w (the *base*);
- a *blossom*: an odd alternating cycle from w to w.



Properties

Properties. In a flower:

- the stem spans $2l + 1$ vertices and contains l matched edges, with $l \geq 0$;
- the blossom B spans $2k + 1$ vertices and contains k matched edges, with $k \geq 1$;
- the base w gets an “even” label from p and it is not matched in B;
- each vertex $i \in B$ is reachable from p along an even path and an odd path:
  - the even path terminates with a matched edge;
  - the odd path terminates with an unmatched edge.

Blossom contraction

All vertices in a blossom are matched.

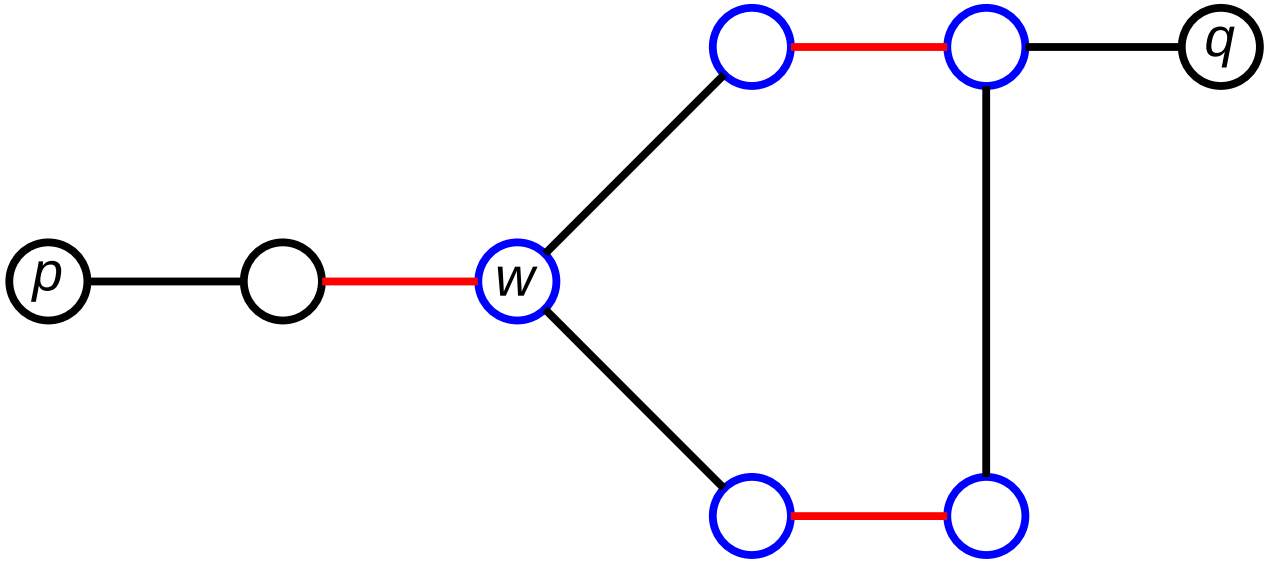
Therefore they can label other vertices out of the blossom only along unmatched edges.

This is possible only if they have an even label.

Assigning all vertices in a blossom an even label, would allow to discover an augmenting path, if one exists.

To achieve this effect, a blossom is contracted into a single pseudo-vertex, which receives an even label.

Example



The algorithm

Every unmatched vertex *p* is tried as a starting vertex of an augmenting path.

To search for an augmenting path from *p*, *p* is labeled “even” and vertices are alternately labeled as ‘odd” and “even” through unmatched and matched edges, respectively.

When a vertex receives two different labels, a blossom has been found: it is contracted into a pseudo-vertex, the pseudo-vertex is labeled “even” and the search resumes.

If an unmatched vertex is labeled “odd”, then an augmenting path has been found: a new matching is obtained, where *p* is matched.

Otherwise, when all vertices have been labeled, the search stops: no augmenting path exists from *p* and *p* is deleted from the graph.

When all vertices are either matched or deleted, the algorithm stops.

Correctness

To prove the correctness of the algorithm, it is necessary and sufficient to prove that blossom contractions do not create or destroy augmenting paths.

Let $\mathcal{G}$ and $\mathcal{G}'$ be the graphs before and after the contraction of B.
Let M and M' be the matchings before and after the contraction of B.

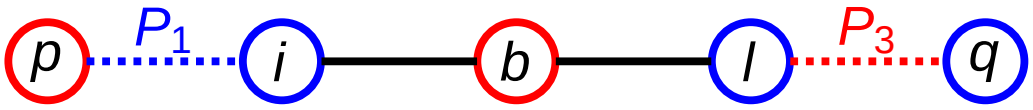
Theorem.

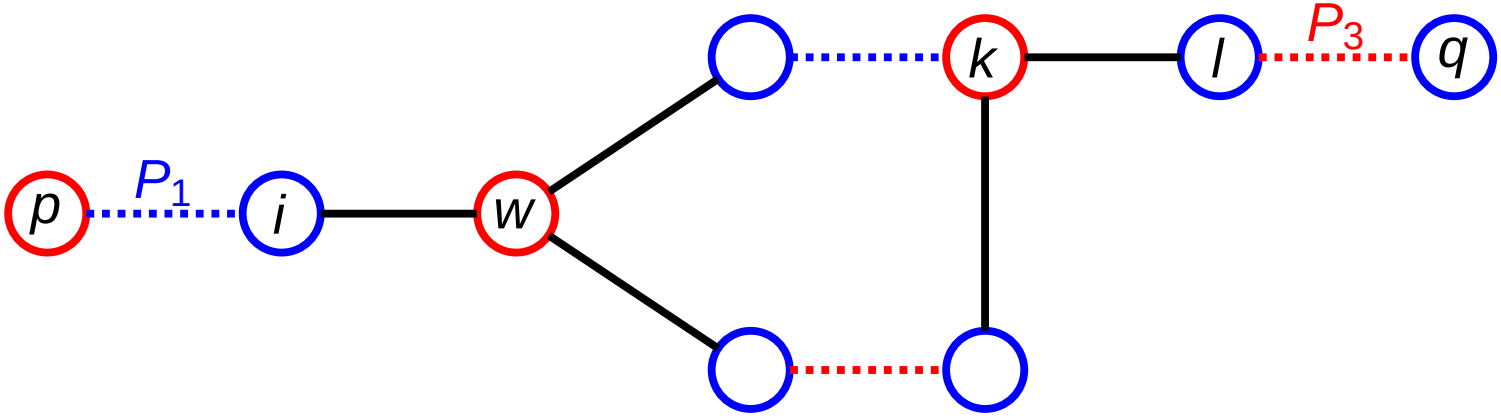
1. If $\mathcal{G}'$ contains an augmenting path P' from vertex p (or the pseudo-vertex containing p) w.r.t. M', then $\mathcal{G}$ contains an augmenting path P from vertex p w.r.t. M.
2. If $\mathcal{G}$ contains an augmenting path P from vertex p to vertex q w.r.t. M, then $\mathcal{G}'$ contains an augmenting path P' from vertex p (or the pseudo-vertex containing p) to q w.r.t. M'.

Correctness

Proof (I). If P' does not contain pseudo-vertices $\Rightarrow P = P'$. $\square$

Otherwise, let b the last pseudo-vertex along P'.



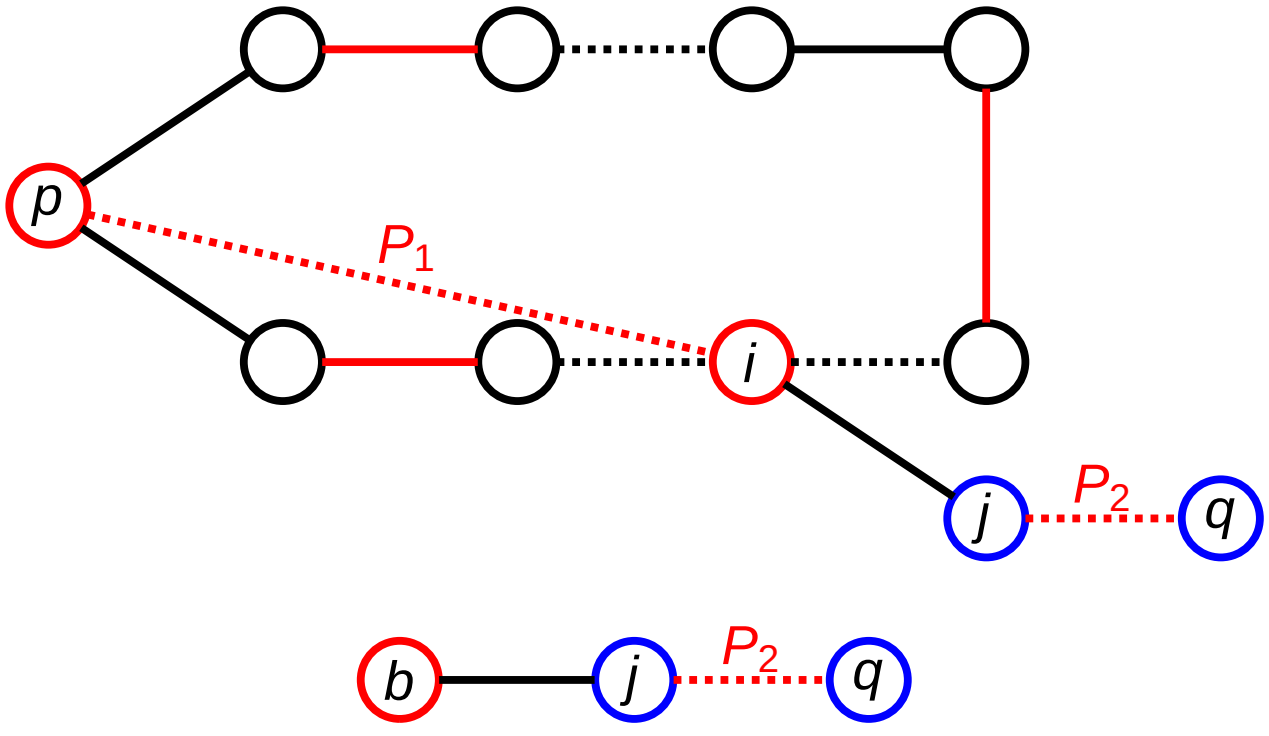


P_2 is the even alternating path from w to k.
$(P_1 \cup [i, w]) \cup P_2 \cup [k, l] \cup P_3$ is an augmenting path in $\mathcal{G}$. $\square$

Correctness

Proof (II). If P has no vertices in common with $B \Rightarrow P' = P$. $\square$

Otherwise, assume B has empty stem: then p is the base of B and b contains p. Let i be the last vertex in B along P.
Then $P = P_1 \cup [i,j] \cup P_2$ for some j with $[i,j] \notin M$.



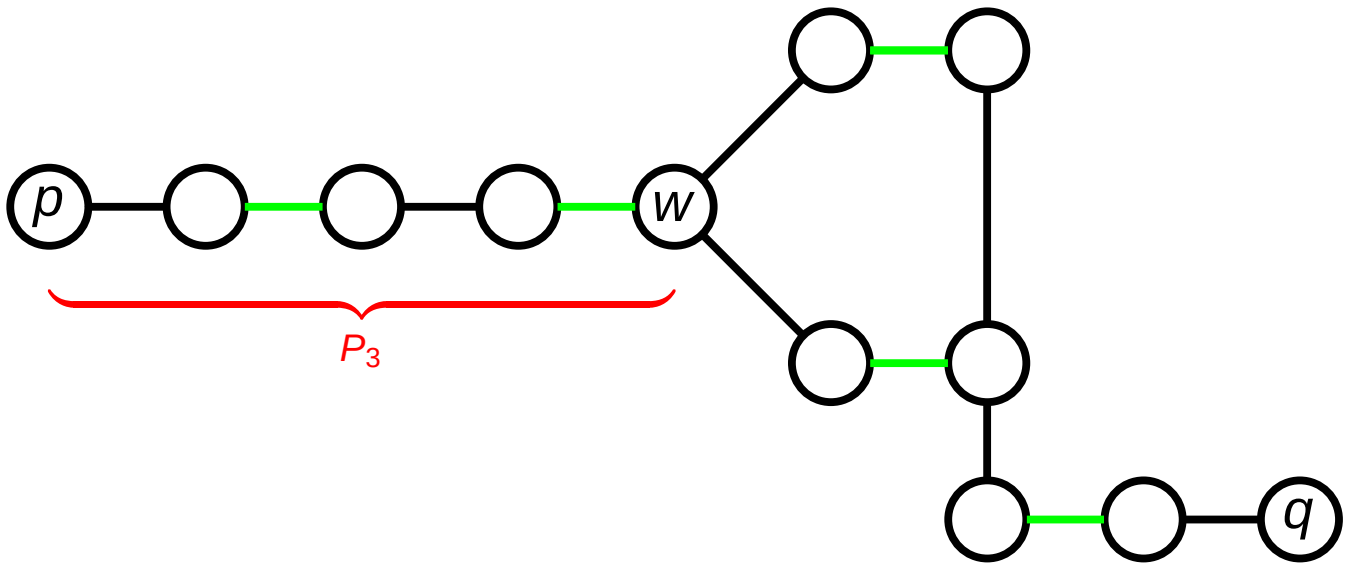
$P' = [b,j] \cup P_2$ is an augmenting path in $\mathcal{G}'$. $\square$

Correctness

Assume now that B has non-empty stem.

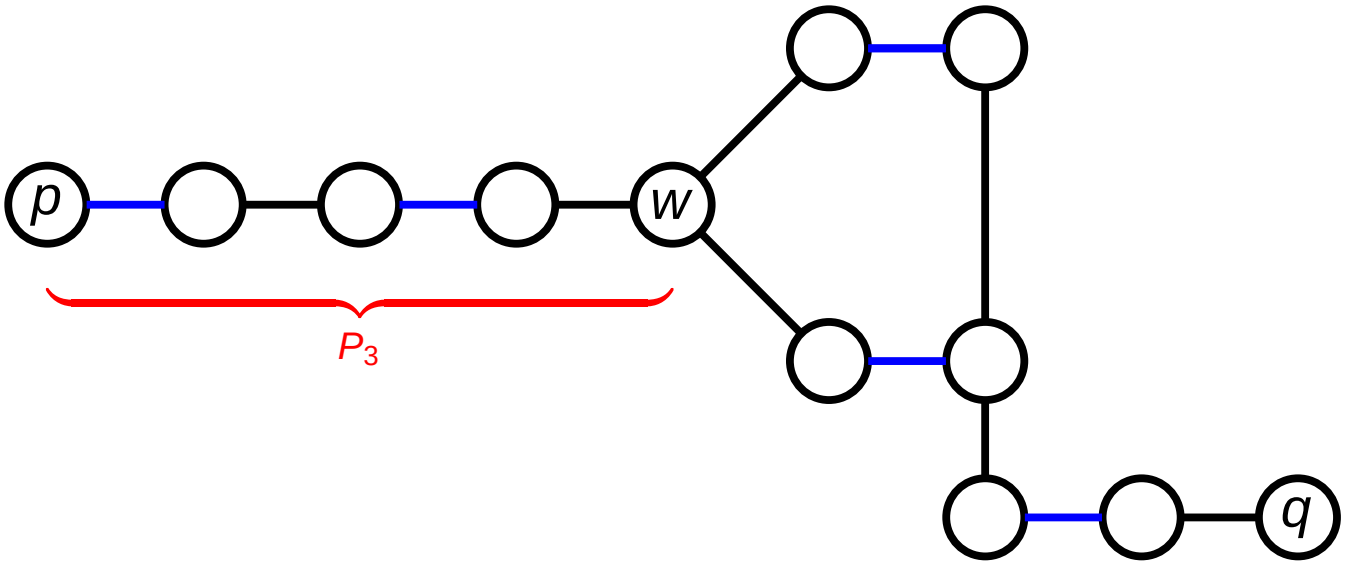
W.l.o.g. we assume that p and q are the only unmatched vertices in $\mathcal{G}$ (the augmenting path P from p to q is not affected by the existence of other unmatched vertices).

Let P_3 be the even alternating path from p to w.



Correctness

Consider the matching $\overline{M} = M \ominus P_3$.



In $\overline{M}$, p is matched and w in unmatched.

Correctness

Since P_3 is even, M and $\overline{M}$ have the same cardinality.

Then, M is not a max matching iff $\overline{M}$ is not a max matching.

By assumption $\mathcal{G}$ contains an augmenting path w.r.t. M.

Therefore $\mathcal{G}$ must contain an augmenting path w.r.t. $\overline{M}$.

But w.r.t. $\overline{M}$ vertices w and q are the only unmatched vertices in $\mathcal{G}$.

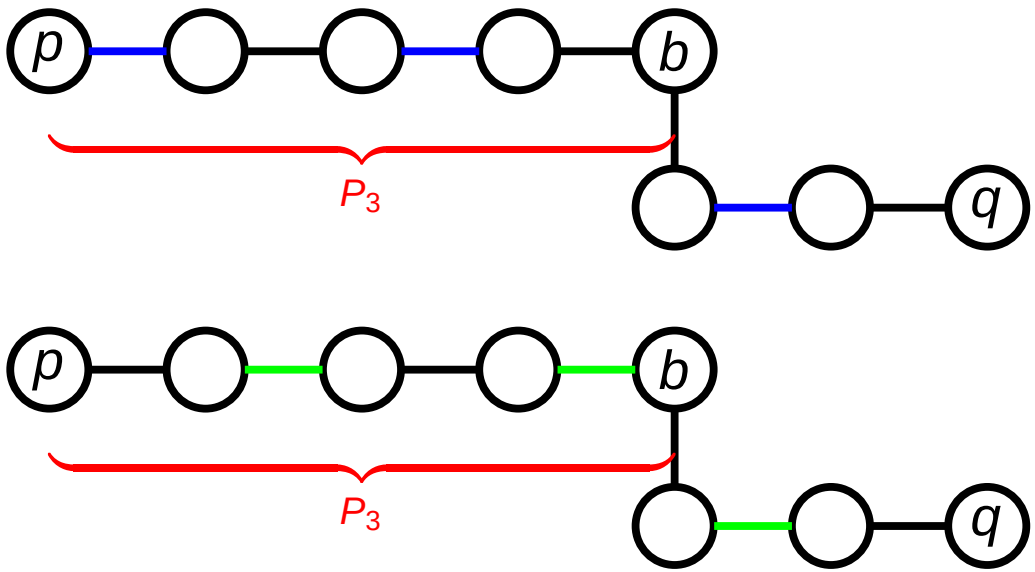
Then $\mathcal{G}$ must contain an augmenting path from w to q.

Correctness

Let $\overline{M}'$ be the matching in $\mathcal{G}'$ corresponding to $\overline{M}$ in $\mathcal{G}$.

In $\overline{M}$, the blossom B has empty stem.

Then the previous proof applies: $\mathcal{G}'$ contains an augmenting path w.r.t. $\overline{M}'$.



Since $\overline{M}'$ and M' have the same cardinality, $\mathcal{G}'$ must also contain an augmenting path w.r.t. M'. □

The algorithm

Input: a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, represented by an adjacency list $E(i)$ for each vertex $i \in \mathcal{V}$ (a linked list).

---

for $i \in \mathcal{V}$ **do**
  $mate(i) = nil$
for $p \in \mathcal{V}$ **do**
  if $mate(p) = nil$ **then**
    $Search(\mathcal{G}, p)$
    if $mate(p) = nil$ **then**
      Delete p from $\mathcal{G}$

$Search(\mathcal{G}, p)$

$\mathcal{V}' \leftarrow \emptyset$
for $i \in \mathcal{V}$ **do**
  $Insert(\mathcal{V}', i)$
  $pseudo(i) \leftarrow false$; $active(i) \leftarrow true$; $label(i) \leftarrow nil$
  $E'(i) \leftarrow E(i)$
$found \leftarrow false$
$label(p) \leftarrow even$
$L \leftarrow \{p\}$
while $(L \neq \emptyset) \wedge (found = false)$ **do**
  $i \leftarrow Extract(L)$
  if $label(i) = even$ **then**
    $ExamineEven(i, found)$
  else
    $ExamineOdd(i, found)$
if $found = true$ **then**
  $Augment$

$ExamineEven(i, found)$

---

for $j \in E'(i)$ **do**
  if $label(j) = even$ **then**
    $Contract(i, j)$
    $Return$
  if $(active(j) = true) \wedge (mate(j) = nil)$ **then**
    $q \leftarrow j$
    $pred(q) \leftarrow i$
    $found \leftarrow true$
    $Return$
  if $(active(j) = true) \wedge (label(j) = nil)$ **then**
    $pred(j) \leftarrow i$
    $label(j) \leftarrow odd$
    $Insert(L, j)$

$ExamineOdd(i, found)$

$j \leftarrow mate(i)$
if $label(j) = odd$ **then**
  $pred(j) \leftarrow i$
  $Contract(i, j)$
  $Return$
if $(label(j) = nil)$ **then**
  $pred(j) \leftarrow i$
  $label(j) \leftarrow even$
  $Insert(L, j)$

$Contract(i,j)$

$B(b) \leftarrow FindBlossom(i,j)$
$Insert(\mathcal{V}', b)$
$pseudo(b) \leftarrow true$
$E'(b) \leftarrow \emptyset$
for $k \in \mathcal{V}$ **do**
  $marked(k) \leftarrow false$
for $j \in B(b)$ **do**
  for $k \in E'(j)$ **do**
    $marked(k) \leftarrow true$
for $k \in \mathcal{V}$ **do**
  if $marked(k) = true$ **then**
    $Insert(E'(b), j)$
    $Insert(E'(j), b)$

$FindBlossom(i, j)$

for $k \in \mathcal{V}$ **do**
    $blossom(k) \leftarrow false$
$k \leftarrow i$; $blossom(k) \leftarrow true$
repeat
    $k \leftarrow pred(k)$; $blossom(k) \leftarrow true$
until $k = p$
$k \leftarrow j$
repeat
    $blossom(k) \leftarrow true$; $k \leftarrow pred(k)$
until $blossom(k) = true$
$w \leftarrow k$
repeat
    $k \leftarrow pred(k)$; $blossom(k) \leftarrow false$
until $k = p$
$B \leftarrow \emptyset$
for $k \in \mathcal{V}$ **do**
    if $blossom(k) = true$ **then**
        $Insert(B, k)$
        $active(k) = false$
return B

Augment

$P \leftarrow \emptyset$
$k \leftarrow q$
repeat
  if $pseudo(pred(k))$ **then**
    $ExpandPred(k)$
  else
    $Insert(P, [pred(k), k])$
    $k \leftarrow pred(k)$
until $k = p$
$M \leftarrow M \ominus P$

$ExpandPred(k)$

$b \leftarrow pred(k)$
for $i \in \mathcal{V}'$ **do**
  $marked(i) \leftarrow false$
for $i \in B(b)$ **do**
  $marked(i) \leftarrow true$
$j \leftarrow FindMarked(E'(k))$
$pred(k) \leftarrow j$

Complexity

The number of augmentations is bounded by $n/2$ since the graph has n vertices and each edge in M matches two of them.

Within each main iteration (search for an augmenting path) the algorithm spends time

1. for contracting blossoms
2. for expanding blossoms
3. for other operations

Each blossom includes at least 3 vertices: then the number of active vertices decreases by at least 2 after each contraction.

Then each main iteration includes at most $n/2$ contractions (and expansions).

The number of pseudo-nodes is bounded by $n/2$ and the cardinality of the adjacency lists by $3n/2$ (still linear in n).

Complexity

1. Time spent for contracting blossoms.

Setting all neighbors of all vertices in the blossom to “unmarked” sums up to $O(n^2)$ over all contractions, because every vertex appears in a blossom at most once.

Updating the adjacency lists for all marked vertices, requires $O(n)$ for each contraction, i.e. $O(n^2)$ over all contractions.

Then, the time spent for contracting is $O(n^2)$ for each main iteration.

Complexity

2. Time spent for expanding blossoms.

In each main iterations no more than $n/2$ calls to *ExpandPred* are required.

Each execution of *ExpandPred* requires $O(n)$.

Then, the time spent for expanding blossoms is $O(n^2)$ for each main iteration.

Complexity

3. Time spent for other operations.

In each main iteration, for each vertex $i \in \mathcal{V}'$

- $active(i)$ is checked from each adjacent vertex: $O(n)$;
- $ExamineOdd(i)$ is executed: $O(1)$;
- $ExamineEven(i)$ is executed: $O(E'(i))$, i.e. $O(n)$ because $|E'(i)| \leq 3n/2$.

Every vertex is examined at most once in a main iteration.

Then, the time spent for other operations is $O(n^2)$ for each main iteration.

Complexity

Within each main iteration the algorithm also

- initializes the graph and the data-structures in $O(n^2)$;
- reconstructs the augmenting path in $O(n)$;
- updates the matching in $O(n)$.

Then, the complexity of each main iteration is $O(n^2)$.

The number of main iterations is $O(n)$.

Therefore the complexity of the algorithm is $O(n^3)$.