Maximum Weighted Matching (II)

Ran Duan

In this lecture

- Maximum weighted matching in general graphs
- Edmonds' algorithm for MWM
- An Application: Christofides algorithm

Review of Hungarian algorithm

- Throughout the algorithm:
  - $y(u)+y(v) \geq w(e) \quad \forall\ e=(u,v)$ (domination)
  - $y(u)+y(v)=w(e)$ if $e \in M$ (tightness)
- Tight edges:
  - An edge $e=(u,v)$ is tight if $y(u)+y(v)=w(e)$
  - Denote the subgraph of tight edges by G_y

Review of Hungarian algorithm

- Let y(u)=N, y(v)=0 (u∈L, v∈R)
- Repeat
    - Augment M in G_y (subgraph of tight edges), until there is no augmenting path any more. (Augmentation step)
    - If M is not perfect, adjust the dual variable y to make more edges tight. (Dual adjustment step)
- Until M is perfect

Dual-adjustment step

- We assign directions to edges in G_y and get G_y’:
    - Non-matching edges: from L to R
    - Matching edges from R to L
    - A path between free vertices of L and R in G_y’ ⇔ An augmenting path in G_y

Dual-adjustment

- In G_y', find the vertices reachable from free vertices of L, call this set Z
  - Since there is no directed path between free vertices of L and R in G_y', Z does not contain free vertices of R

- Let $y(u)=y(u)-\Delta$ for $u \in L \cap Z$
- Let $y(v)=y(v)+\Delta$ for $v \in R \cap Z$
  - Δ can bring more tight edges without breaking the domination condition
  - For integer-weighted graph, we can set $\Delta=1$

An example

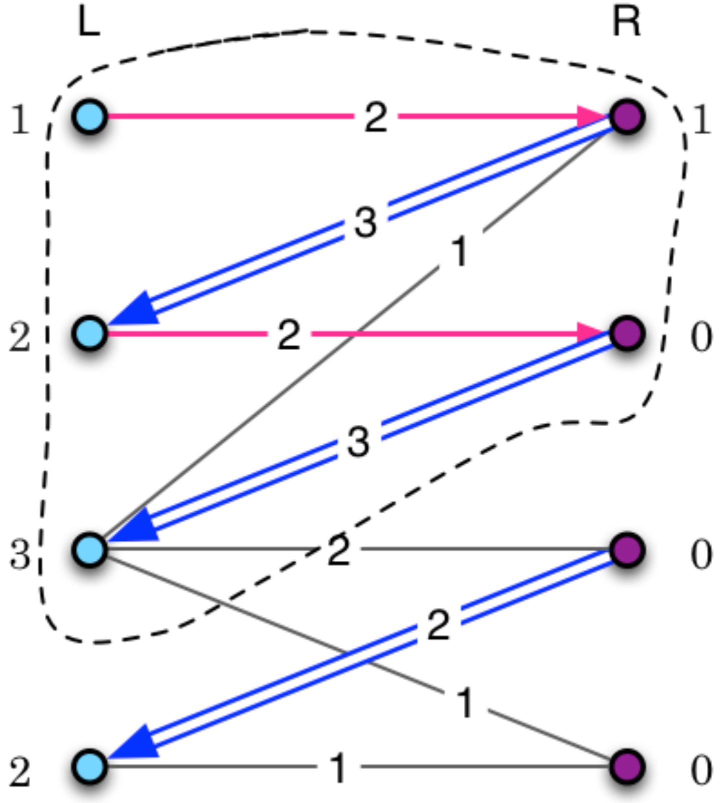
- Tight edges
- Matching edges

(Dual adjustment step)

Let Z be the set of vertices reachable from free vertices of L

Let y(u)=y(u)-Δ for u∈L∩Z

Let y(v)=y(v)+Δ for v∈R∩Z



An example

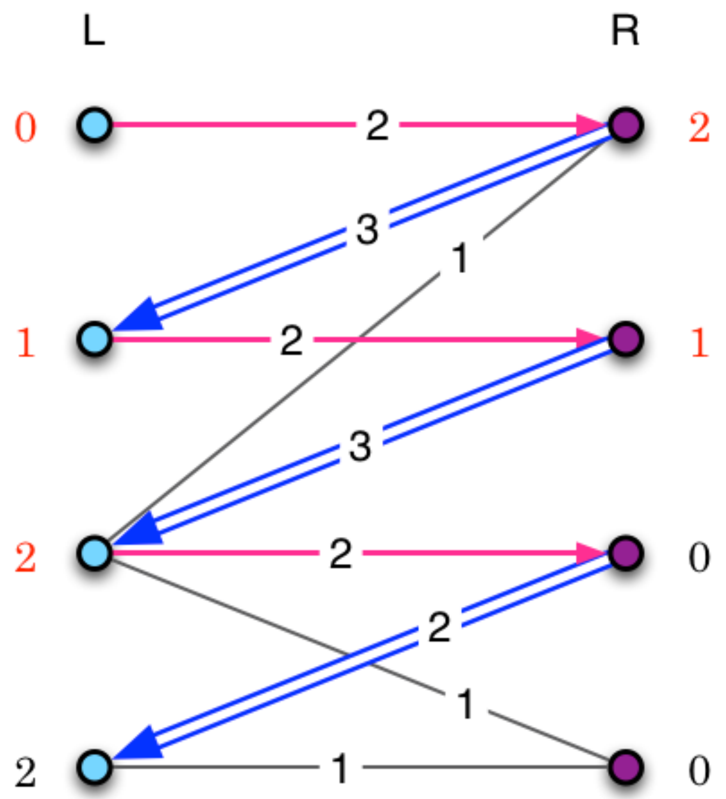
- Tight edges
- Matching edges

(Dual adjustment step)

Let Z be the set of vertices reachable from free vertices of L

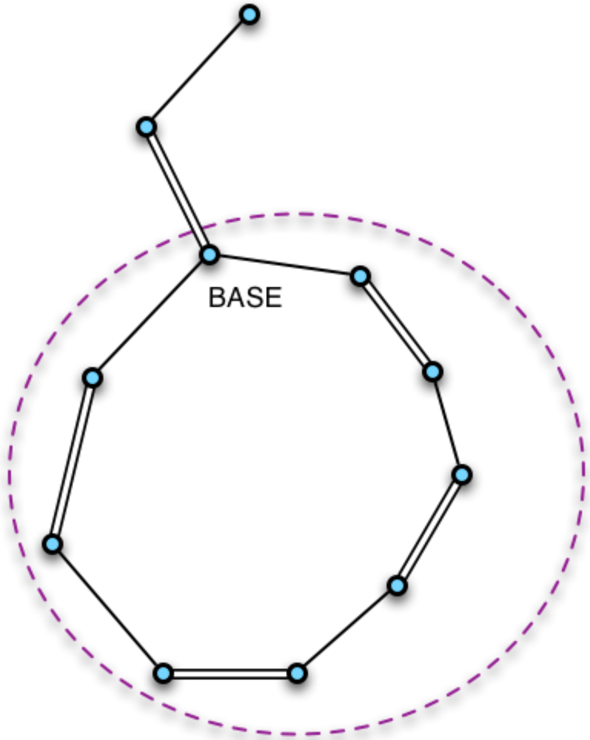
Let y(u)=y(u)-Δ for u∈L∩Z

Let y(v)=y(v)+Δ for v∈R∩Z



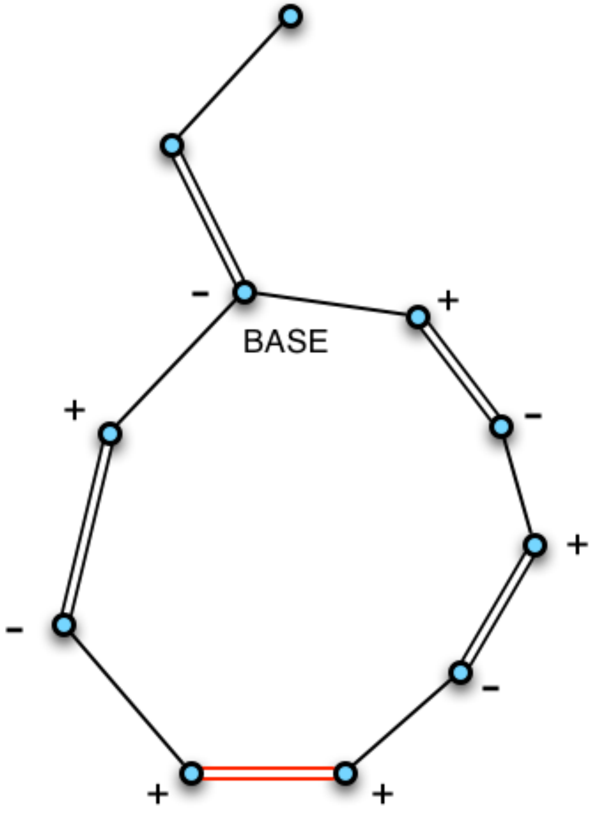
Blossom

- A blossom B is a cycle in G consisting of 2k + 1 edges of which exactly k belong to M.
- The only vertex whose matching edge is not in B is called the base.



Difficulty for general graphs

- Since a blossom contains odd number of vertices, it hard to add some amount to some vertices and subtract the same amount to others and keep all the edges of the blossom tight.



Difficulty for general graphs

- Since a blossom contains odd number of vertices, it’s hard to add some amount to some vertices and subtract the same amount to others and keep all the edges of the blossom tight.
- Thus it cannot guarantee Z to be larger every time.

Solution: Dual-variables on blossoms

- Edmond's primal-dual algorithm:

$$y : Vertices \rightarrow \Re$$

$$z : Blossom \rightarrow \Re, z \geq 0$$

$$yz(u,v) = y(u) + y(v) + \sum_{u,v \in B} z(B)$$

Solution: Dual-variables on blossoms

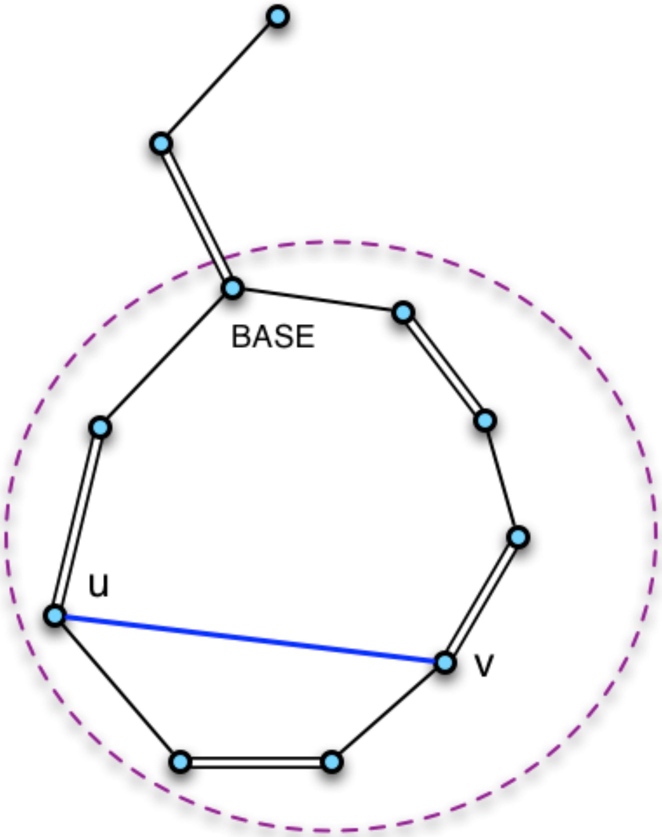
- Edmond's primal-dual algorithm:

$$y : Vertices \rightarrow \Re$$

$$z : Blossom \rightarrow \Re, z \geq 0$$

$$yz(u,v) = y(u) + y(v) + \sum_{u,v \in B} z(B)$$

u,v both in the blossom B,
not necessarily a blossom edge



Solution: Dual-variables on blossoms

- Edmond's primal-dual algorithm:

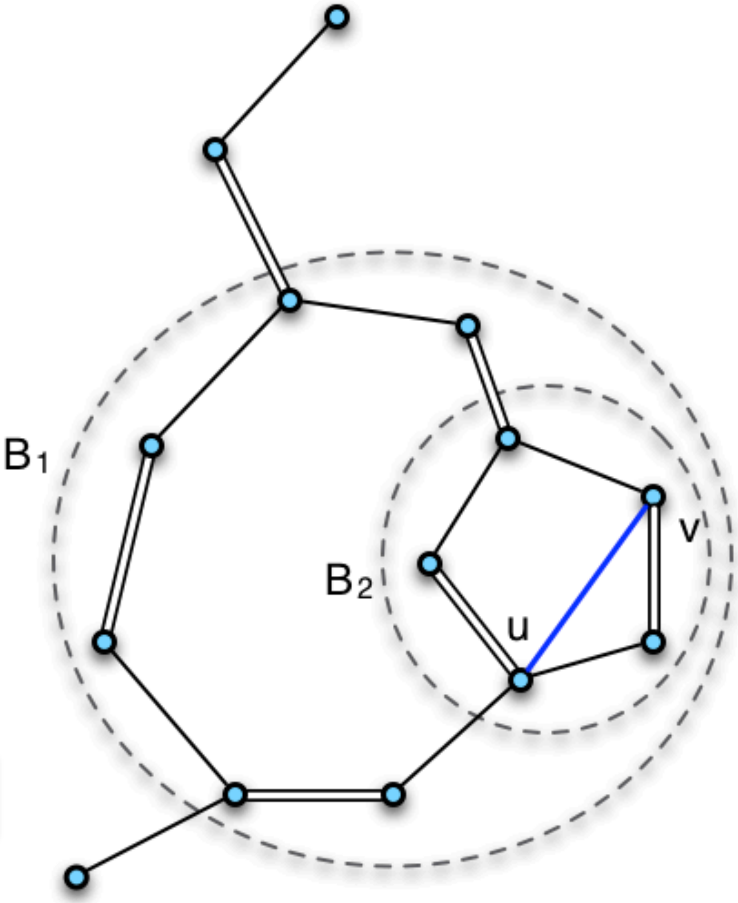
$$y : Vertices \rightarrow \Re$$

$$z : Blossom \rightarrow \Re, z \geq 0$$

$$yz(u,v) = y(u) + y(v) + \sum_{\substack{u,v \in B \\ B \in \Omega}} z(B)$$

They can be in multiple blossoms.
Define the set of all the blossoms to be Ω

Here B_1 is a root blossom.



Solution: Dual-variables on blossoms

- Edmond's primal-dual algorithm:

$$y : Vertices \to \Re$$

$$z : Blossom \to \Re, z \geq 0$$

$$yz(u,v) = y(u) + y(v) + \sum_{u,v \in B} z(B)$$

- They will satisfy:
  - z(B)≥0 for all blossoms B, and z(B)>0 if B is a root blossom
  - yz(e)≥w(e) for all edges e
  - yz(e)=w(e) if e is a matching edge or a blossom edge

Solution: Dual-variables on blossoms

- Edmond's primal-dual algorithm:

$$y : Vertices \rightarrow \Re$$

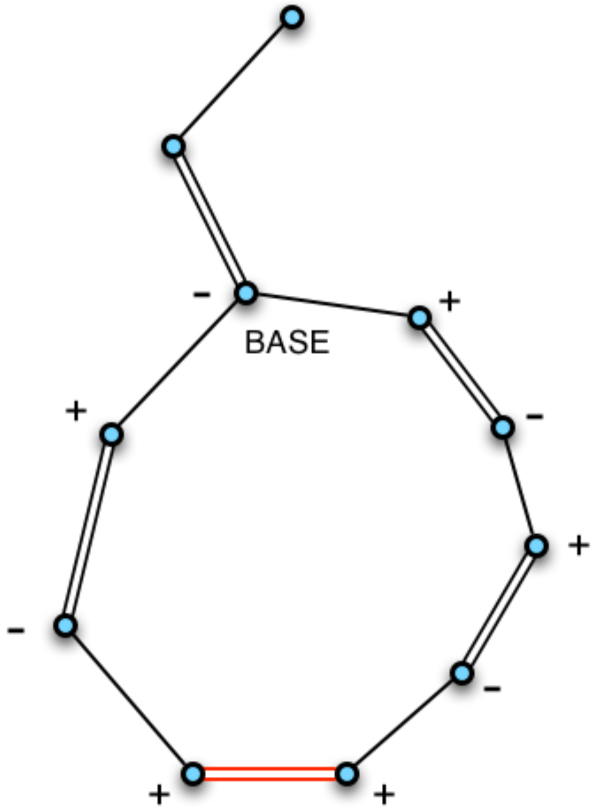
$$z : Blossom \rightarrow \Re, z \geq 0$$

$$yz(u,v) = y(u) + y(v) + \sum_{u,v \in B} z(B)$$

- They will satisfy:
  - z(B)≥0 for all blossoms B, and z(B)>0 if B is a root blossom
  - yz(e)≥w(e) for all edges e
  - yz(e)=w(e) if e is a matching edge or a blossom edge
  - (So all blossom edges are tight)

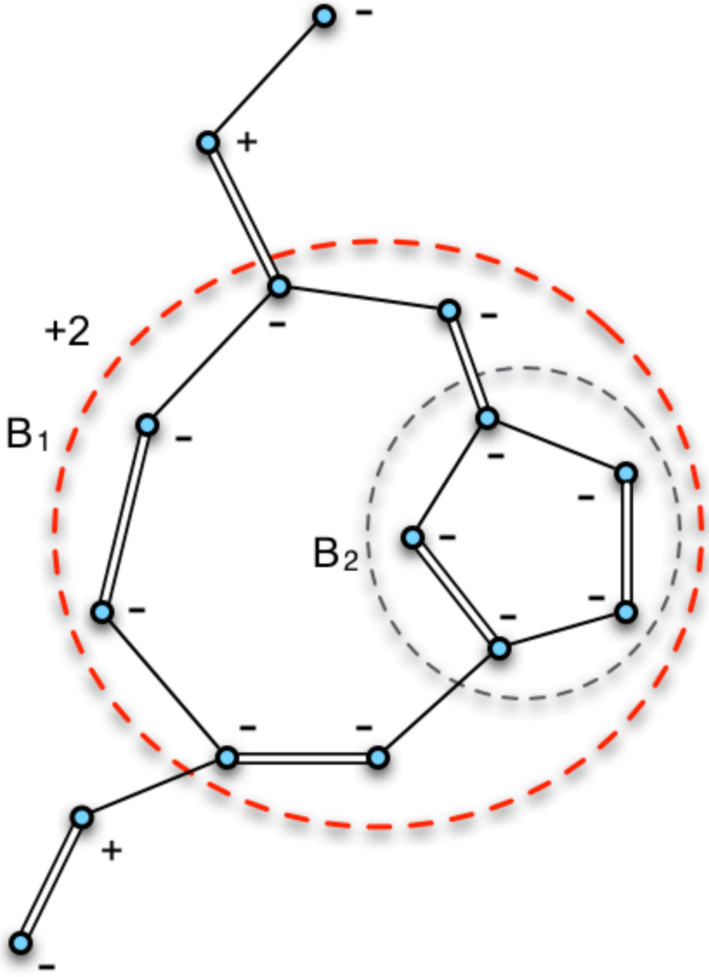
Why we need z-value?

- Remind the difficulty:



Why we need z-value?

- Now we can just minus (plus) Δ to all the vertices in the blossom, and plus (minus) 2Δ to the z-value of the root blossom.
- Then $yz(e)$ for edges in the blossom will remain unchanged.



Solution: Dual-variables on blossoms

- Edmond’s primal-dual algorithm:

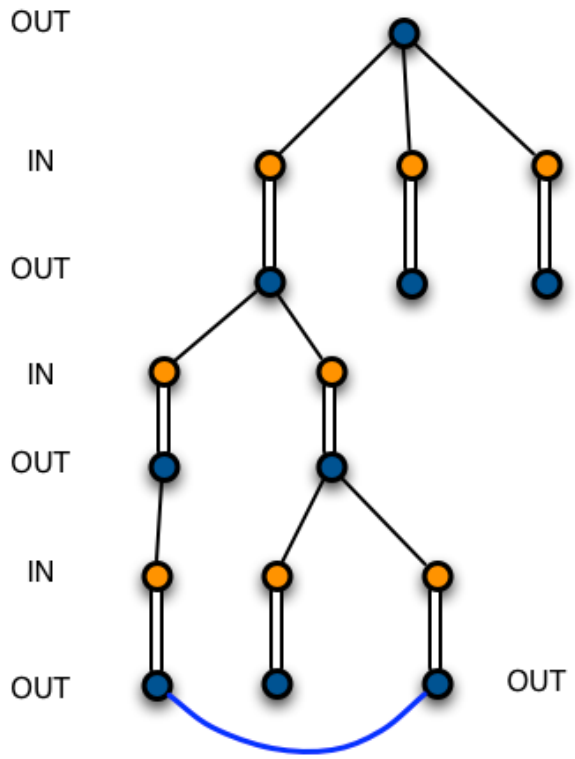
$$y : Vertices \rightarrow \Re$$

$$z : Blossom \rightarrow \Re, z \geq 0$$

$$yz(u,v) = y(u) + y(v) + \sum_{u,v \in B} z(B)$$

- Eligible edges:
    - Matching edges
    - Blossom edges
    - Other edges satifying yz(e)=w(e)

- Remind next Monday we talked about alternating trees and we marked vertices to be EVEN or ODD based on their lengths of alternating path to free vertices
- This time we mark them OUT and IN instead.



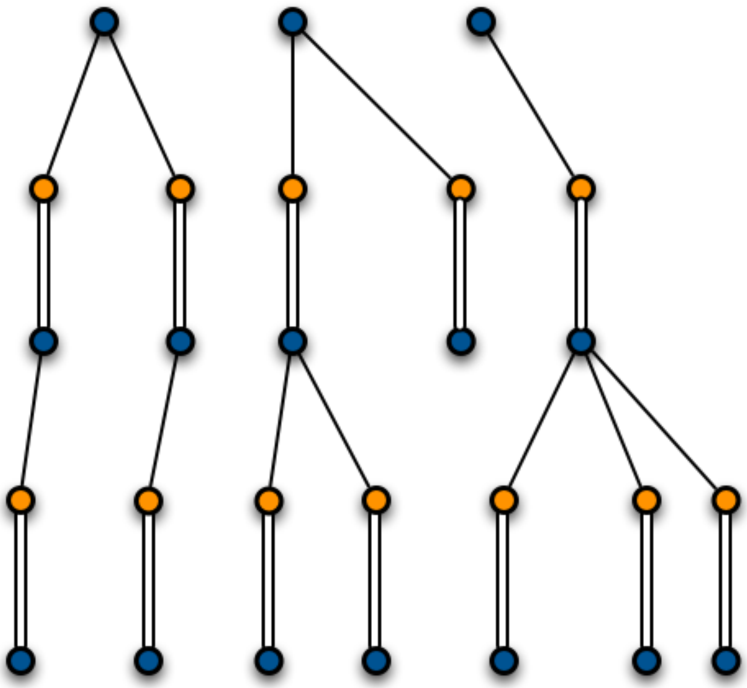
Procedure (Every iteration)

- (There is contracted graph G’ which contracts all blossoms)
- Run breath-first search from all free vertices on eligible edges
- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
    - An augmenting path: update the matching M, blossoms and G’
    - New blossom: Shrinking the blossom B, z(B):=0, update G’
- Otherwise run the dual-adjustment:
    - For vertex v’ in G’ which have been labeled OUT or IN, we labeled OUT or IN to the original vertex v in G contained in v’.
    - $y(v):=y(v)-\Delta$ for all OUT vertex v
    - $y(v):=y(v)+\Delta$ for all IN vertex v
    - $z(B):=z(B)+2\Delta$ if B is a root blossom containing OUT vertices
    - $z(B):=z(B)-2\Delta$ if B is a root blossom containing IN vertices
    - Dissolve root blossoms with zero z-values (non-root blossoms can have zero z-values), update G’ and set of eligible edges

Procedure (Every iteration)

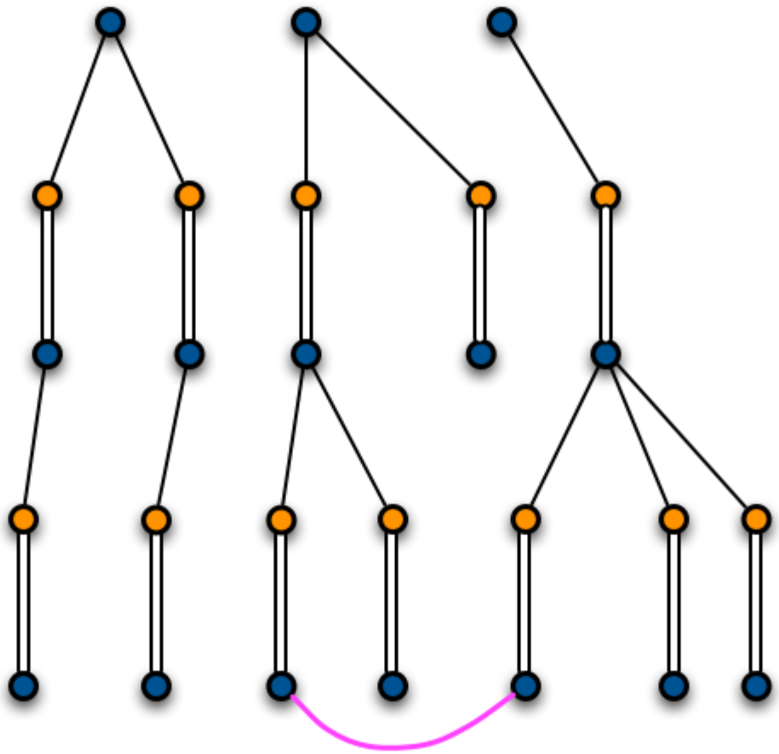
- (There is contracted graph G’ which contracts all blossoms)
- Run breath-first search from all free vertices on eligible edges
- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
  - An augmenting path: update the matching M, blossoms and G’
  - New blossom: Shrinking the blossom B, $z(B):=0$, update G’
- Otherwise run the dual-adjustment:
  - For vertex v’ in G’ which have been labeled OUT or IN, we labeled OUT or IN to the original vertex v in G contained in v’.
  - $y(v):=y(v)-\Delta$ for all OUT vertex v
  - $y(v):=y(v)+\Delta$ for all IN vertex v
  - $z(B):=z(B)+2\Delta$ if B is a root blossom containing OUT vertices
  - $z(B):=z(B)-2\Delta$ if B is a root blossom containing IN vertices
  - Dissolve root blossoms with zero z-values (non-root blossoms can have zero z-values), update G’ and set of eligible edges

- (There is contracted graph G’ which contracts all blossoms)
- Run breath-first search from all free vertices on eligible edges
- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom



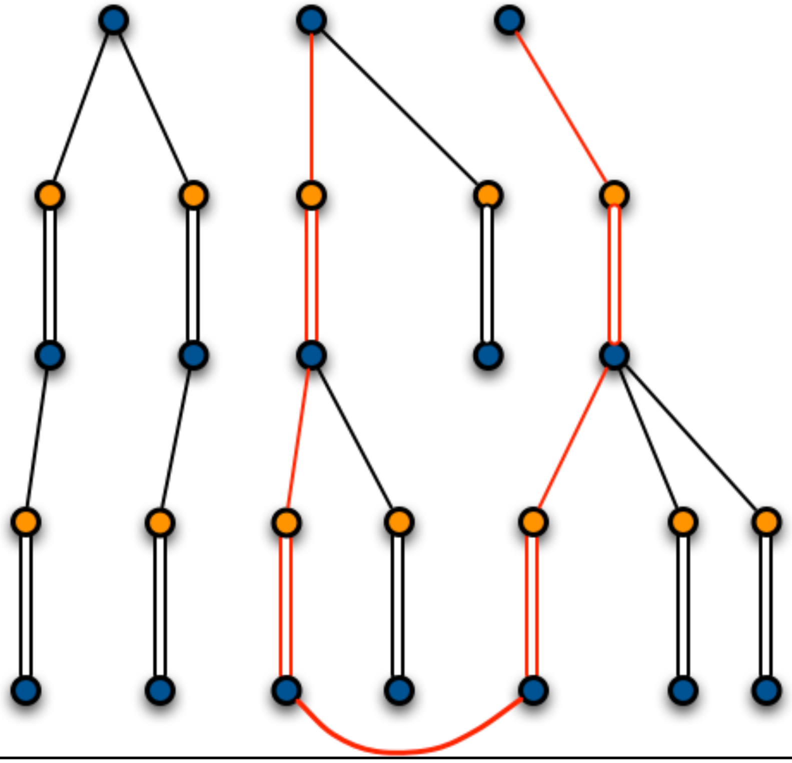
OUT
IN

- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
- If the edge connects different trees from different free vertices, then we can get an augmenting path



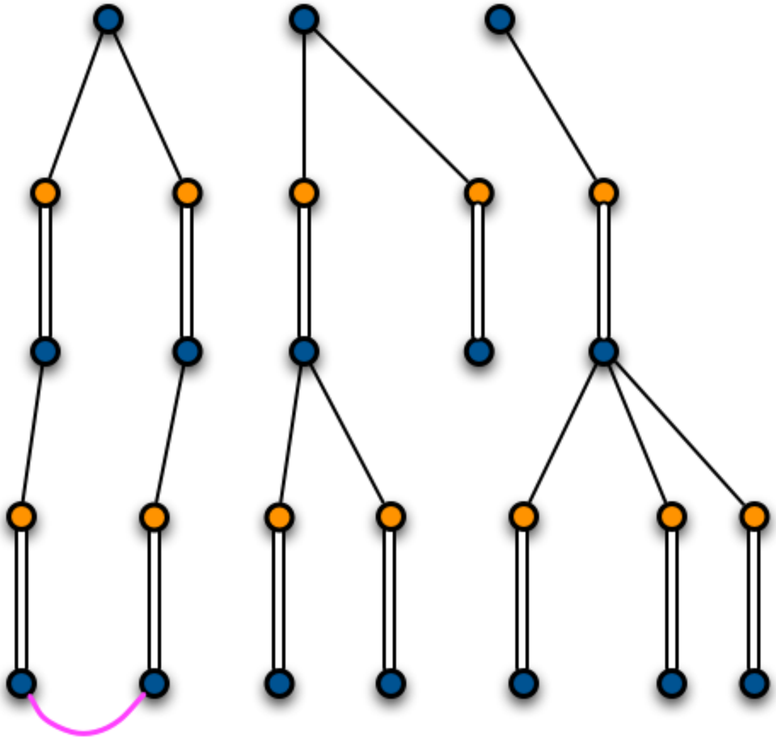
OUT
IN

- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
- If the edge connects different trees from different free vertices, then we can get an augmenting path



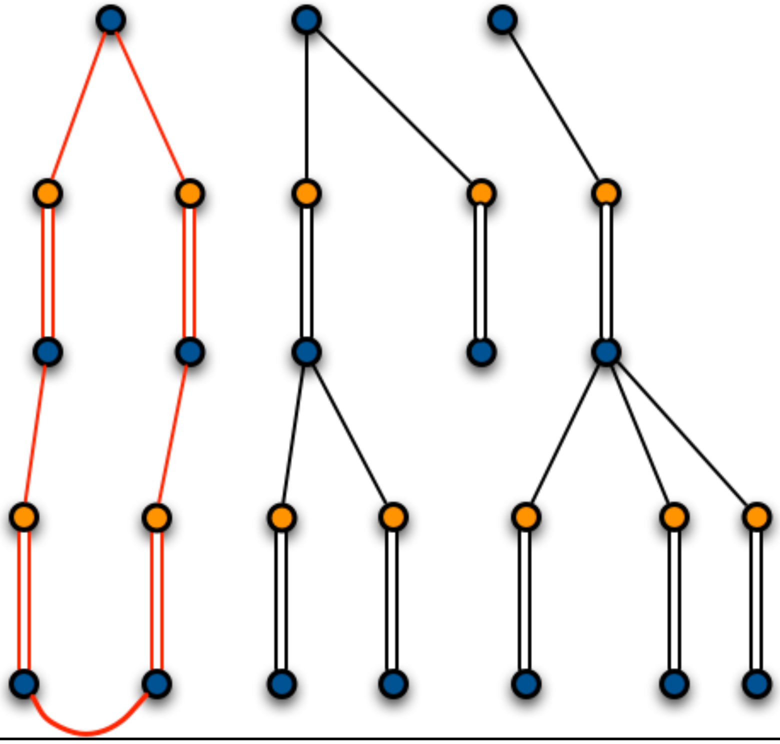
OUT
IN

- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
- If the edge connects vertices from the same free vertex, then we can get a new blossom



OUT
IN

- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
- If the edge connects vertices from the same free vertex, then we can get a new blossom



OUT
IN

Procedure (Every iteration)

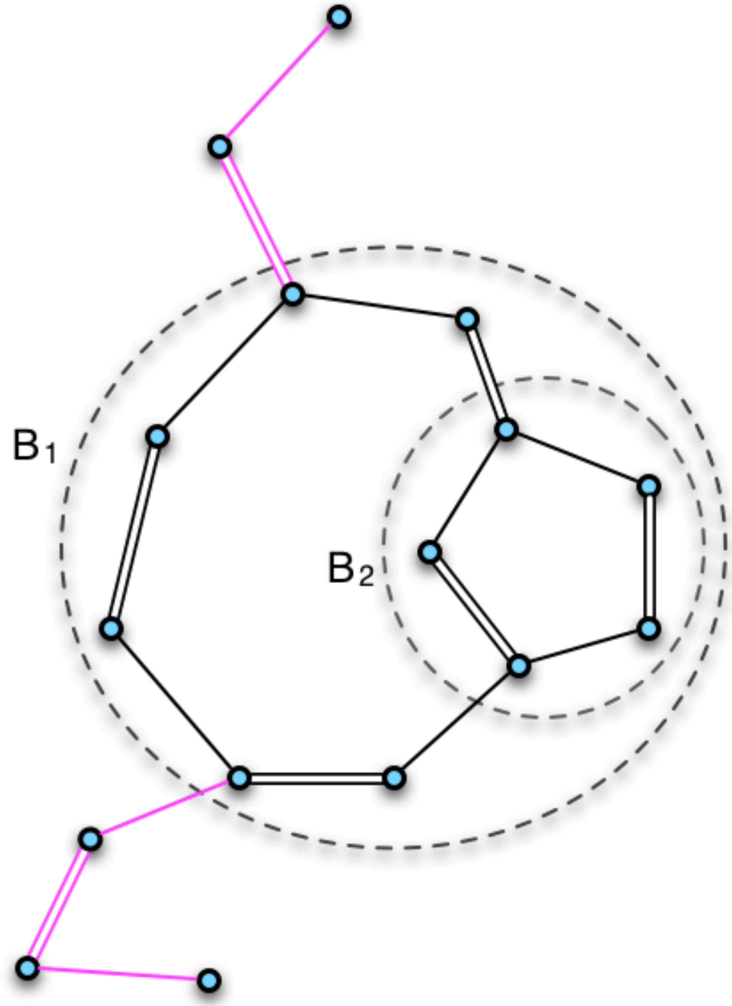
- (There is contracted graph G’ which contracts all blossoms)
- Run breath-first search from all free vertices on eligible edges
- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
  - An augmenting path: update the matching M, blossoms and G’
  - New blossom: Shrinking the blossom B, z(B):=0, update G’
- Otherwise run the dual-adjustment:
  - For vertex v’ in G’ which have been labeled OUT or IN, we labeled OUT or IN to the original vertex v in G contained in v’.
  - $y(v):=y(v)-\Delta$ for all OUT vertex v
  - $y(v):=y(v)+\Delta$ for all IN vertex v
  - $z(B):=z(B)+2\Delta$ if B is a root blossom containing OUT vertices
  - $z(B):=z(B)-2\Delta$ if B is a root blossom containing IN vertices
  - Dissolve root blossoms with zero z-values (non-root blossoms can have zero z-values), update G’ and set of eligible edges

Augmentation

- Since a blossom can have positive z-value, we cannot dissolve all the blossoms.
- On the augmenting path, switch between non-matching and matching in G’ and in every blossom
- So for every blossom on the augmenting path, the base will change.

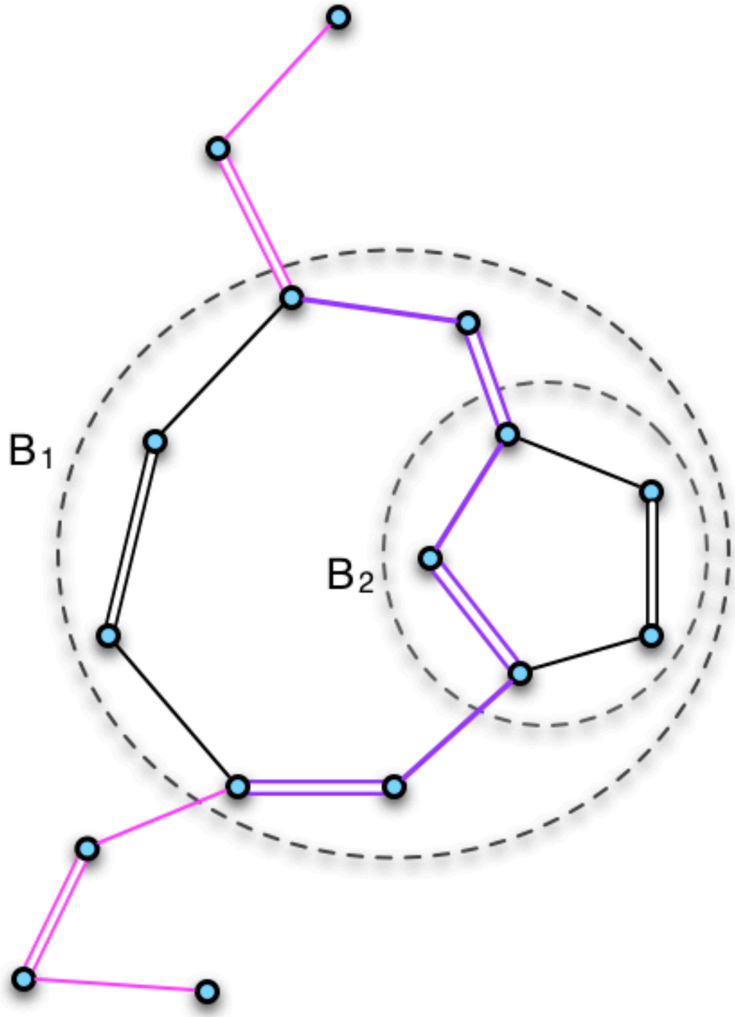
For example

- It's an augmenting path in G'



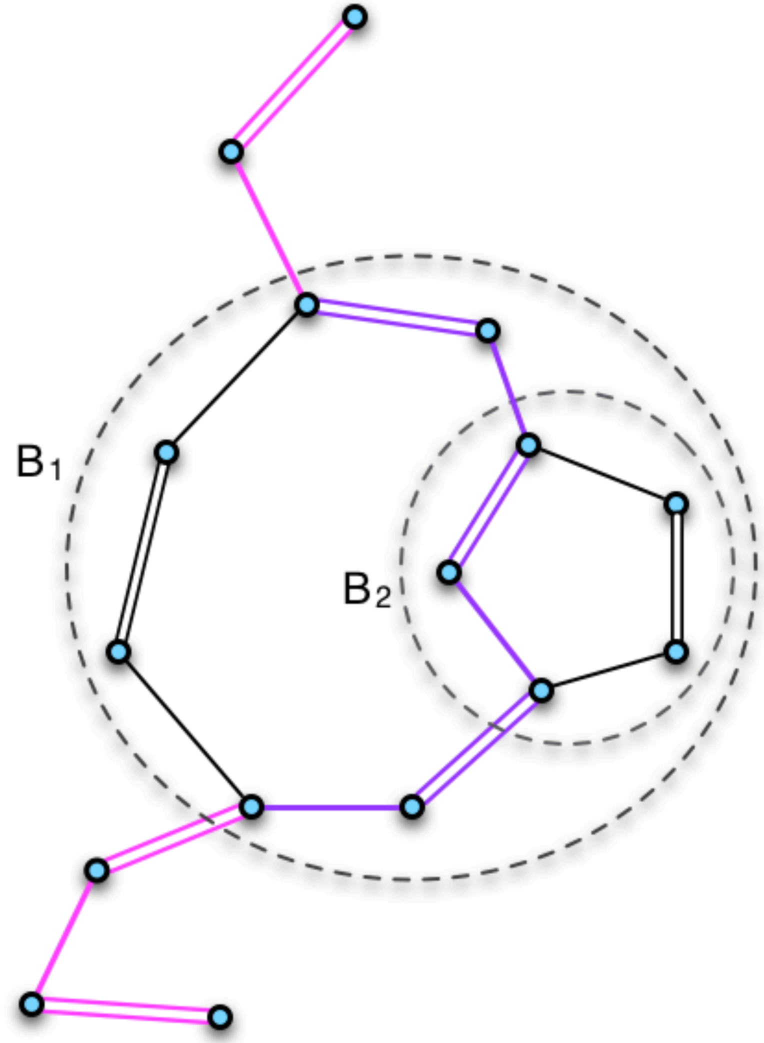
For example

- This is the corresponding augmenting path in the real graph G

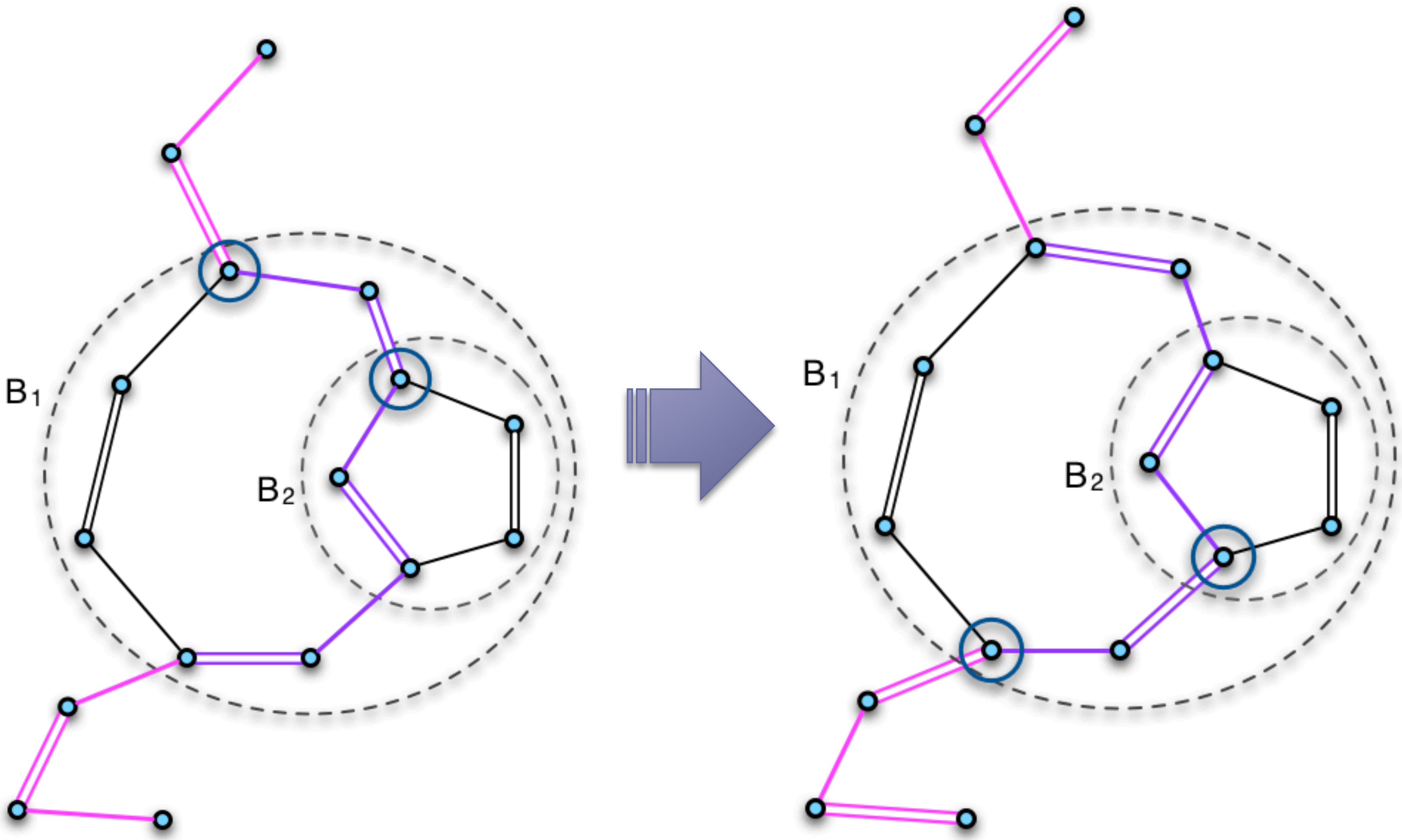


For example

- After augmentation



- After augmentation
- The base of B_1 and B_2 change



Augmentation

- On the augmenting path, switch between non-matching and matching in G’ and in every blossom
- So for every blossom on the augmenting path, the base will change.

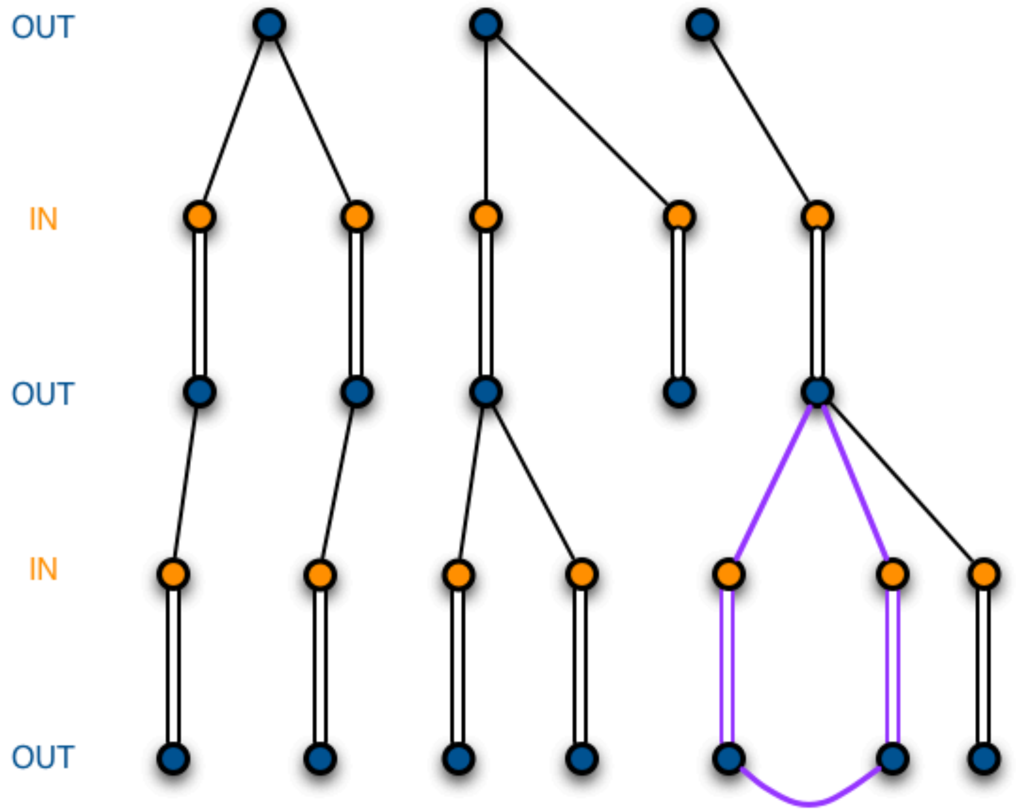
- After augmentation, functions y,z and blossoms do not change, so all tight edges (including all matching edges and blossom edges) remain tight.
- We need to dissolve all the blossoms of zero z-value.

Procedure (Every iteration)

- (There is contracted graph G' which contracts all blossoms)
- Run breath-first search from all free vertices on eligible edges
- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
  - An augmenting path: update the matching M, blossoms and G'
  - New blossom: Shrinking the blossom B, z(B):=0, update G'
- Otherwise run the dual-adjustment:
  - For vertex v' in G' which have been labeled OUT or IN, we labeled OUT or IN to the original vertex v in G contained in v'.
  - $y(v):=y(v)-\Delta$ for all OUT vertex v
  - $y(v):=y(v)+\Delta$ for all IN vertex v
  - $z(B):=z(B)+2\Delta$ if B is a root blossom containing OUT vertices
  - $z(B):=z(B)-2\Delta$ if B is a root blossom containing IN vertices
  - Dissolve root blossoms with zero z-values (non-root blossoms can have zero z-values), update G' and set of eligible edges

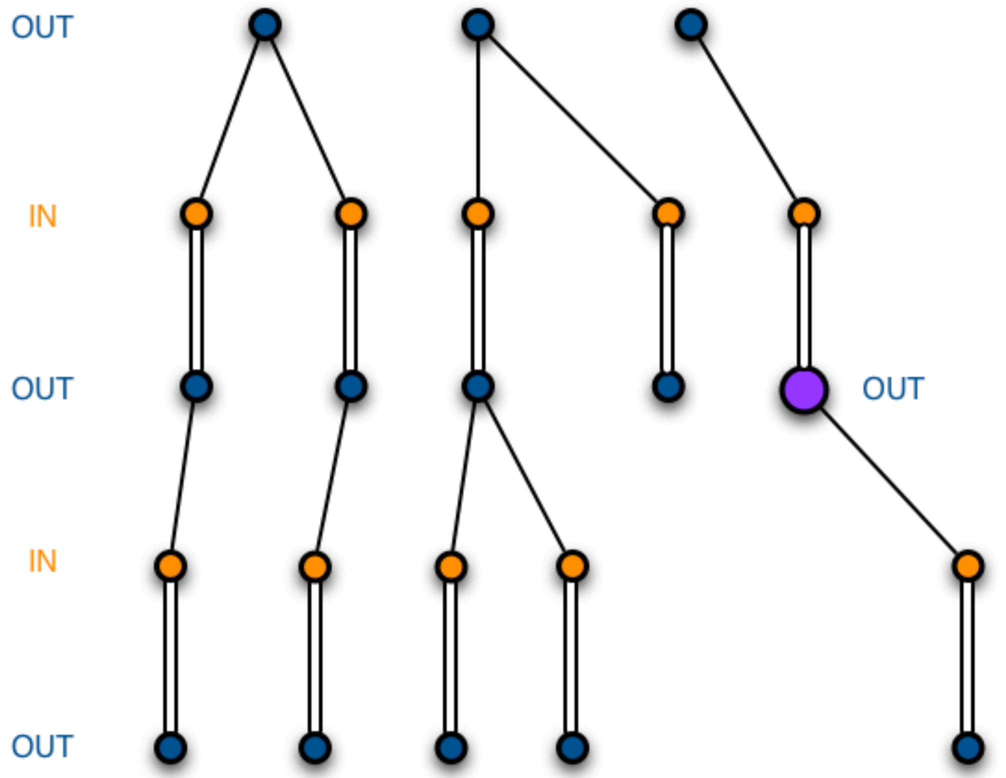
Blossoms Shrinking

- When we find such a blossom,



Blossoms Shrinking

- When we find such a blossom, shrink it to a single vertex.



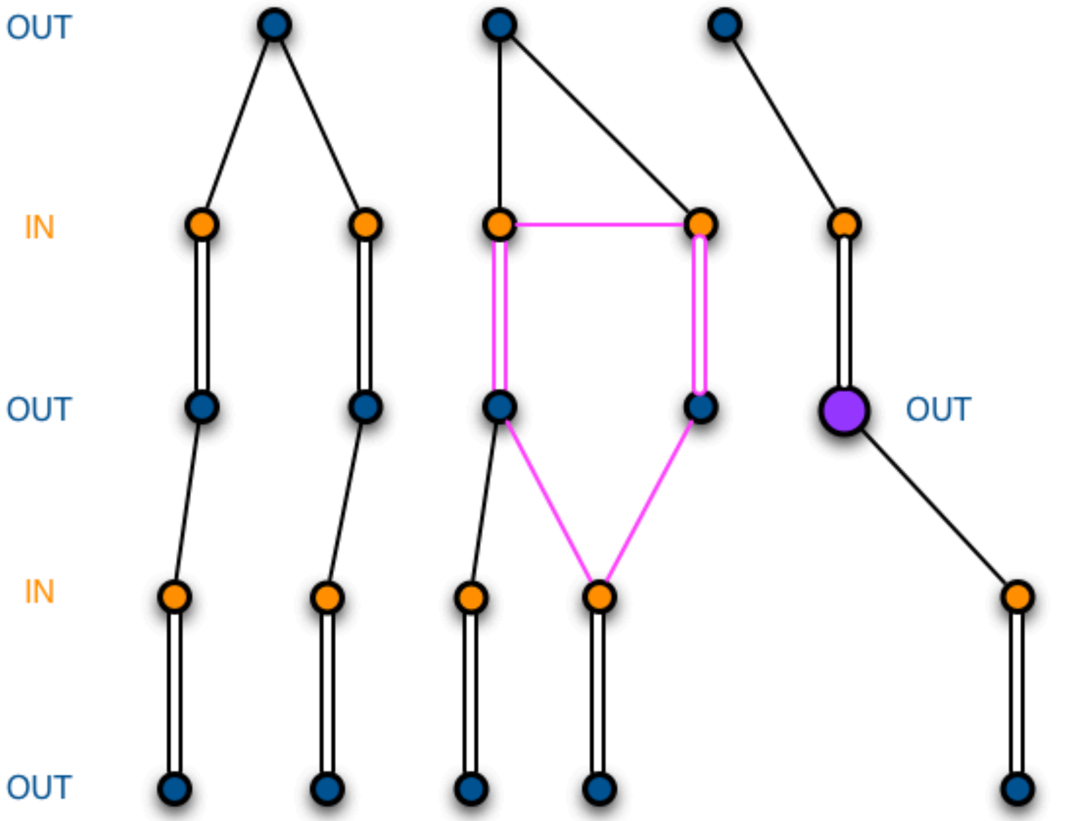
Blossoms Shrinking

- Important Property: all the new blossoms in this algorithm have EVEN length augmenting path to free vertices. (OUT blossoms)

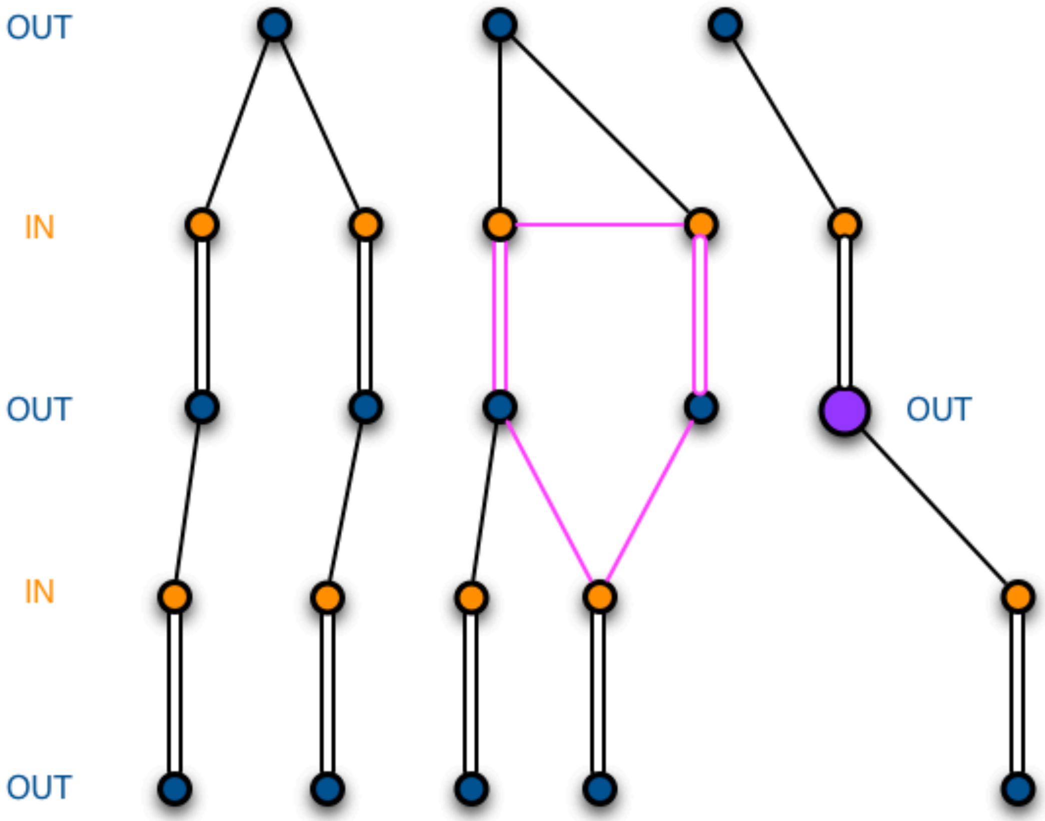
- Since new IN blossoms cannot be discovered

- Important Property: all the new blossoms in this algorithm have EVEN length augmenting path to free vertices. (OUT blossoms)
- Since new IN blossoms cannot be discovered

- This would be an IN blossom:



- But we do not scan such non-matching edges from IN vertices



Blossoms Shrinking

- Important Property: all the new blossoms in this algorithm have EVEN length augmenting path to free vertices. (OUT blossoms)

- Since new IN blossoms cannot be discovered
- Also, all root IN blossoms with zero z-value have been dissolved after augmentation or dual-adjustment.

- These can guarantee that there is no IN root blossoms with zero z-value

Procedure (Every iteration)

- (There is contracted graph G’ which contracts all blossoms)
- Run breath-first search from all free vertices on eligible edges
- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
  - An augmenting path: update the matching M, blossoms and G’
  - New blossom: Shrinking the blossom B, $z(B):=0$, update G’
- Otherwise run the dual-adjustment:
  - For vertex v’ in G’ which have been labeled OUT or IN, we labeled OUT or IN to the original vertex v in G contained in v’.
  - $y(v):=y(v)-\Delta$ for all OUT vertex v
  - $y(v):=y(v)+\Delta$ for all IN vertex v
  - $z(B):=z(B)+2\Delta$ if B is a root blossom containing OUT vertices
  - $z(B):=z(B)-2\Delta$ if B is a root blossom containing IN vertices
  - Dissolve root blossoms with zero z-values (non-root blossoms can have zero z-values), update G’ and set of eligible edges

- (There is contracted graph G’ which contracts all blossoms)
- Run breath-first search from all free vertices on eligible edges
- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
  - An augmenting path: update the matching M, blossoms and G’
  - New blossom: Shrinking the blossom B, z(B):=0, update G’
- Otherwise run the dual-adjustment:
  - For vertex v’ in G’ which have been labeled OUT or IN, we labeled OUT or IN to the original vertex v in G contained in v’.
  - y(v):=y(v)-Δ for all OUT vertex v
  - y(v):=y(v)+Δ for all IN vertex v
  - z(B):=z(B)+2Δ if B is a root blossom containing OUT vertices
  - z(B):=z(B)-2Δ if B is a root blossom containing IN vertices
  - Dissolve root blossoms with ze z-values (non-root blossoms can have zero z-values), update G’ and s r eligible edges

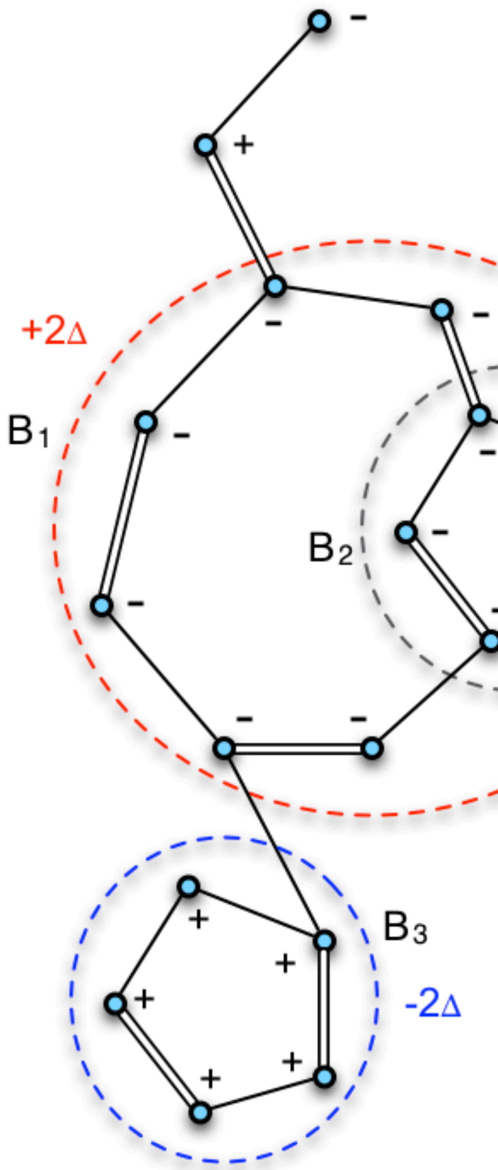
No IN root blossoms with zero z-value

- (There is contracted graph G’ which contracts all blossoms)
- Run breath-first search from all free vertices on eligible edges
- If we find an edge connecting OUT vertices, there is an augmenting path or a new blossom
  - An augmenting path: update the matching M, blossoms and G’
  - New blossom: Shrinking the blossom B, z(B):=0, update G’
- Otherwise run the dual-adjustment:
  - For vertex v’ in G’ which have been labeled OUT or IN, we labeled OUT or IN to the original vertex v in G contained in v’.
  - y(v):=y(v)-½ for all OUT vertex v
  - y(v):=y(v)+½ for all IN vertex v
  - z(B):=z(B)+1 if B is a root blossom containing OUT vertices
  - z(B):=z(B)-1 if B is a root blossom containing IN vertices
  - Dissolve root blossoms with zero z-values (non-root blossoms can have zero z-values), update G’ and s[illegible]r eligible edges

For simplicity we let Δ=½ for integer-weighted graphs

Dual-adjustment

- Here B_1 is an OUT root blossom and B_3 is an IN root blossom



Correctness of dual-adjustment

- After searching for augmenting paths and blossoms, in G' there doesn't exist the following:
    - IN root blossoms with zero z-value
    - tight edge connecting OUT vertices

    - tight edge connecting an OUT vertex and an unmarked vertex

Correctness of dual-adjustment

- After searching for augmenting paths and blossoms, in G’ there doesn’t exist the following:
  - IN root blossoms with zero z-value
  - tight edge connecting OUT vertices
    - Otherwise we can find an augmenting path or an OUT blossom
  - tight edge connecting an OUT vertex and an unmarked vertex
    - Otherwise we can mark that vertex “IN”

Remind: Dual-variables on blossoms

- Edmond's primal-dual algorithm:

$$y : Vertices \rightarrow \Re$$

$$z : Blossom \rightarrow \Re, z \geq 0$$

$$yz(u,v) = y(u) + y(v) + \sum_{u,v \in B} z(B)$$

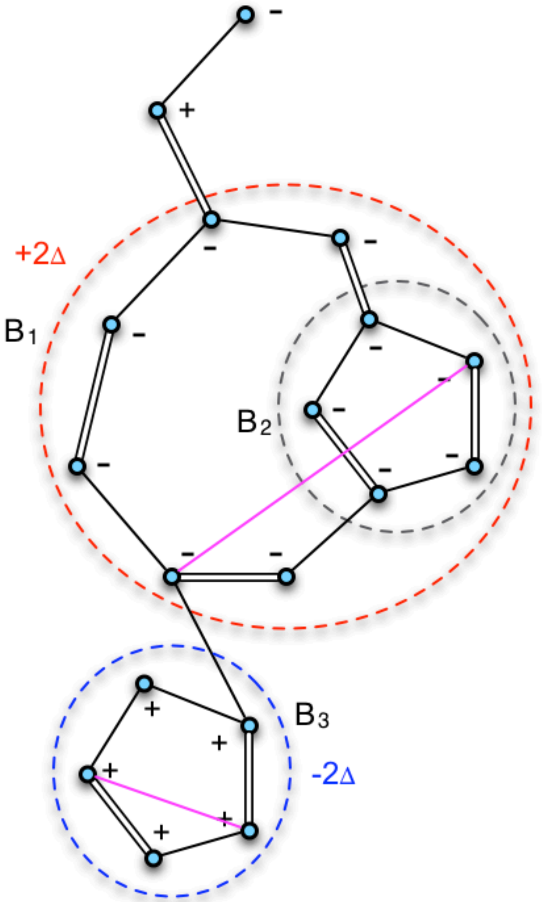
- They will satisfy:
  - z(B)≥0 for all blossoms B, and z(B)>0 if B is a root blossom
  - yz(e)≥w(e) for all edges e
  - yz(e)=w(e) if e is a matching edge or a blossom edge
  - (So all blossom edges are tight)

Correctness

$y(v):=y(v)-\frac{1}{2}$ for all OUT vertex v
$y(v):=y(v)+\frac{1}{2}$ for all IN vertex v
$z(B):=z(B)+1$ if B is an OUT root blossom
$z(B):=z(B)-1$ if B is an IN root blossom

- All the y-values of vertices are integer multiples of ½
- All the z-values of blossoms are integers

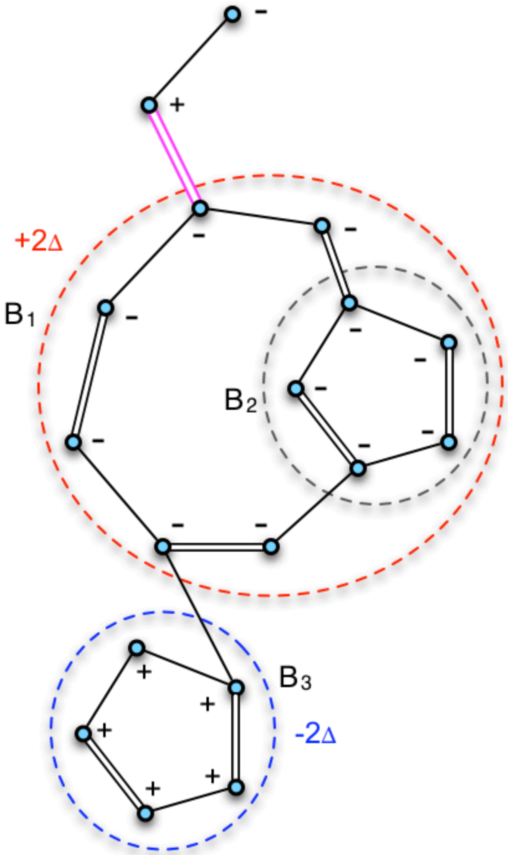
- For each edge e in a blossom, the yz(e) will not change
  - If an OUT blossom, y(u),y(v) decrease by ½, z(B) increases by 1
  - If an IN blossom, y(u),y(v) increase by ½, z(B) decreases by 1
  - If it is not marked, nothing change



Correctness

$y(v):=y(v)-\frac{1}{2}$ for all OUT vertex v
$y(v):=y(v)+\frac{1}{2}$ for all IN vertex v
$z(B):=z(B)+1$ if B is an OUT root blossom
$z(B):=z(B)-1$ if B is an IN root blossom

- All the y-values of vertices are integer multiples of ½
- All the z-values of blossoms are integers
- For matching edges not in a blossom, one end is OUT and one end is IN, also yz(e) will not change.
- Thus, tightness for blossom edges and matching edges still holds

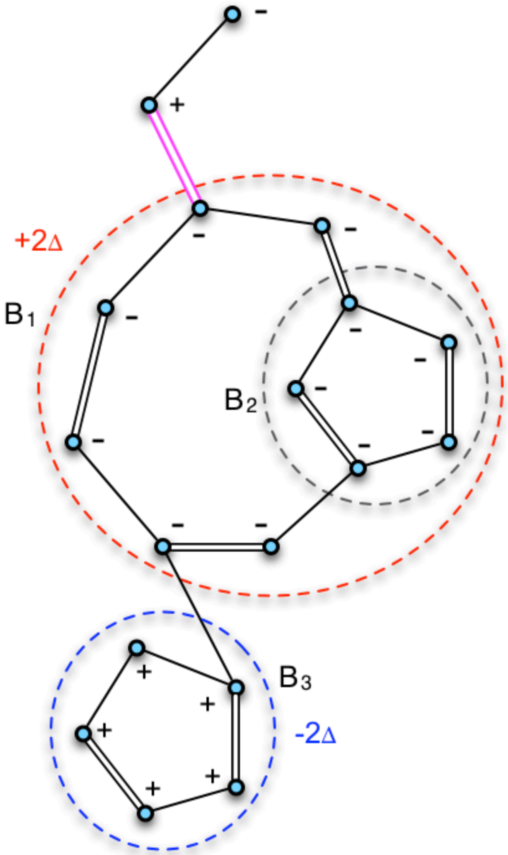


Correctness

y(v):=y(v)-½ for all OUT vertex v
y(v):=y(v)+½ for all IN vertex v
z(B):=z(B)+1 if B is an OUT root blossom
z(B):=z(B)-1 if B is an IN root blossom

- z(B)≥0 for all blossoms B, and z(B)>0 if B is a root blossom

- All new root blossoms are OUT, so we add their z-value to 1 just after contracting them.
- We dissolve all the root blossoms with zero z-values after augmentation and dual-adjustment.



Correctness

- Domination: yz(e)≥w(e)
- We have already prove it when e=(u,v) is contained in a blossom
- Otherwise yz(e)=y(u)+y(v), and it can only decrease when both are OUT or one is OUT and one is unmarked. (e cannot be tight in these cases)

Correctness

y(v):=y(v)-½ for all OUT vertex v
y(v):=y(v)+½ for all IN vertex v
z(B):=z(B)+1 if B is an OUT root blossom
z(B):=z(B)-1 if B is an IN root blossom

- All the y-values of vertices are integer multiples of ½
- All the z-values of blossoms are integers

- Define the parity of y(v) for a vertex v to be the parity of the its multiple of ½
- (For example, 3/2 is an odd multiple of ½)

Correctness

y(v):=y(v)-½ for all OUT vertex v
y(v):=y(v)+½ for all IN vertex v
z(B):=z(B)+1 if B is an OUT root blossom
z(B):=z(B)-1 if B is an IN root blossom

- All the y-values of vertices are integer multiples of ½
- All the z-values of blossoms are integers

- Then for tight edges (u,v), since

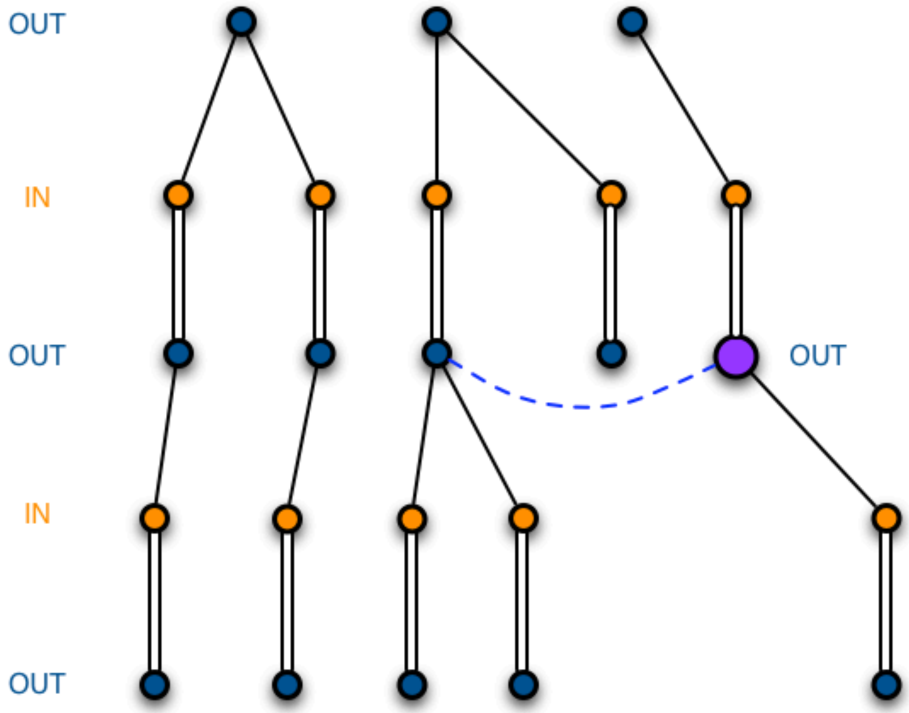
$$w(u,v) = yz(u,v) = y(u) + y(v) + \sum_{u,v \in B} z(B)$$

- and w(u,v) is an integer, so y(u) and y(v) have the same parity (to the multiples of ½).

- Since in the search, all OUT and IN vertices are linked by tight edges, so they have the same parity.

- When e is not a matching or blossom edges,
- yz(e)=y(u)+y(v), and it can only decrease when both are OUT or one is OUT and one is unmarked. (e cannot be tight in these cases)

- If both u and v are OUT, u,v have the same parity, so y(u)+y(v)>w(e) is an integer, so y(u)+y(v)≥w(e)+1.
- After dual-adjustment, y(u)+y(v)≥w(e)



y(v):=y(v)-½	for all OUT vertex v
y(v):=y(v)+½	for all IN vertex v
z(B):=z(B)+1	if B is OUT
z(B):=z(B)-1	if B is IN

- When e is not a matching or blossom edges,
- yz(e)=y(u)+y(v), and it can only decrease when both are OUT or one is OUT and one is unmarked. (e cannot be tight in these cases)

- If both u and v are OUT, u,v have the same parity, so y(u)+y(v)>w(e) is an integer, so y(u)+y(v)≥w(e)+1.
- After dual-adjustment, y(u)+y(v)≥w(e)

- If only u is OUT, yz(e) only decrease by ½, so y(u)+y(v)≥w(e) after dual-adjustment.

Correctness

- We have show that dual-adjustments will keep the properties of dual-variables:
    - $z(B) \geq 0$ for all blossoms B, and $z(B) > 0$ if B is a root blossom
    - $yz(e) \geq w(e)$ for all edges e
    - $yz(e) = w(e)$ if e is a matching edge or a blossom edge
    - (So all blossom edges are tight)

- Also, if at the beginning all vertices have y-value N/2,
- all free vertices have the same and smallest y-value throughout the algorithm

- Until we get a perfect matching

When achieving a perfect matching M*…

- Since all matching edges are tight, and every blossom B contains (|B|-1)/2 matching edges.

$$w(M^*) = \sum_{e \in M^*} yz(e) = \sum_{u \in V} y(u) + \sum_{B \in \Omega} z(B) \cdot \frac{(|B|-1)}{2}$$

- For every other perfect matching M, a blossom B cannot contain more than (|B|-1)/2 edges of M, so

$$w(M) \le \sum_{e \in M} yz(e) = \sum_{u \in V} y(u) + \sum_{B \in \Omega} z(B) |M \cap E(B)| \le \sum_{u \in V} y(u) + \sum_{B \in \Omega} z(B) \cdot \frac{(|B|-1)}{2}$$

- (Here E(B) is the set of all edges whose both ends are in B)
- So M* is a maximum weighted perfect matching

- Also, we can find a maximum weighted matching when the y-values of free vertices reach 0.

Analysis of running time

- Find the minimum difference yz(e)-w(e) of non-tight edges e
- Then do the dual-adjustment until some new edges becomes tight
- So in O(m) time, these two kinds of new tight edges may emerge:
    - Tight edge connecting two OUT vertices,
      then we can find a new blossom
    - Tight edge connecting an OUT vertex and an unmarked vertex,
      then the set of marked vertices will be larger

Analysis of running time

- Find the minimum difference yz(e)-w(e) of non-tight edges e
- Then do the dual-adjustment until some new edges becomes tight
- So in O(m) time, these two kinds of new tight edges may emerge:
    - Tight edge connecting two OUT vertices, then we can find a new blossom
    - Tight edge connecting an OUT vertex and an unmarked vertex, then the set Z of marked vertices will be larger

- Before we find an augmenting path, we can find O(n) blossoms, and the set Z can become larger O(n) times
- Also finding a blossom takes O(m) time
- So the running time for one augmenting path is O(mn)

Analysis of running time

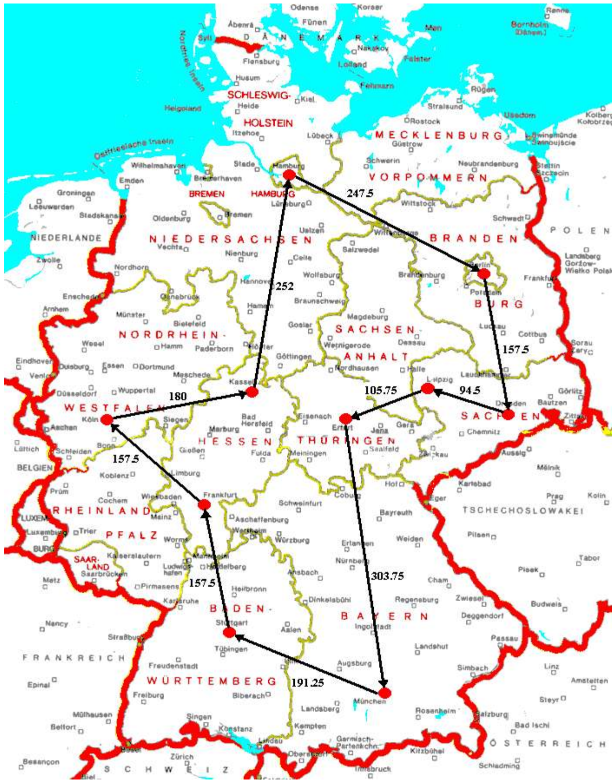
- So the time for one augmenting path is O(mn)
- Since there are O(n) augmenting paths
- The total running time is O(mn^2)

An application of MWM in general graphs

- **Christofides heuristic algorithm**
  - For an approximate solution of the traveling salesman problem
  - On a complete graph G=(V,E,w), and its edge weights satisfy the triangle inequality:
    - $w(x,y)+w(y,z) \geq w(x,z)$, for all $x,y,z \in V$

Traveling salesman problem

- Find the shortest cycle which visits every node exactly once.



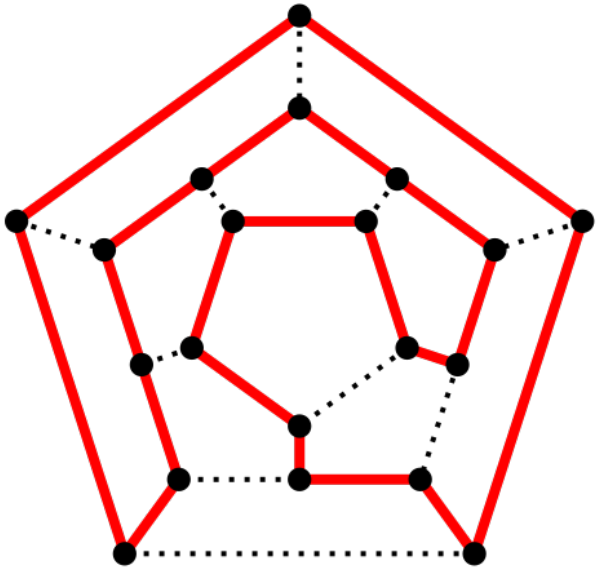
Traveling salesman problem

- Find the shortest cycle which visits every node exactly once.
- NP-complete: one of the hardest problem in NP to find a polynomial time algorithm.

- Thus, we need polynomial time approximate algorithms.

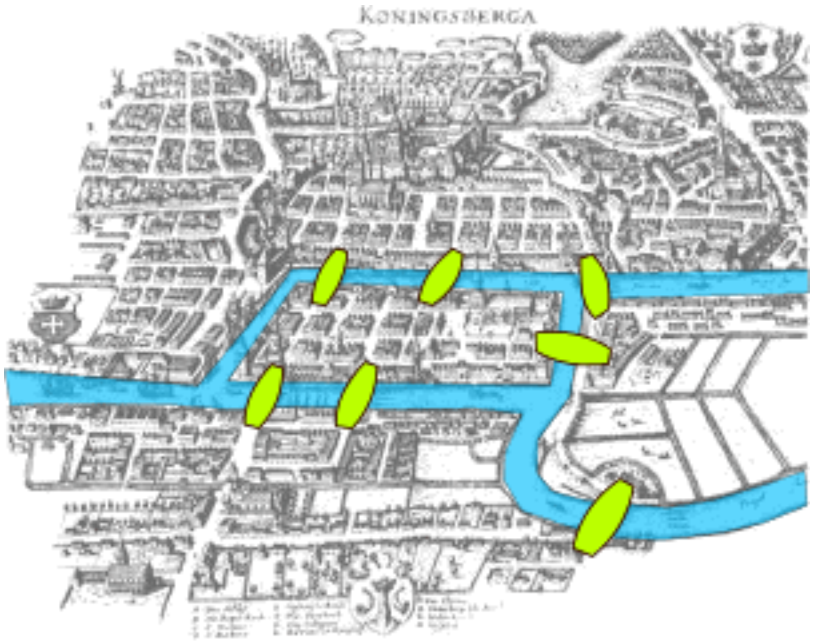
Hamilton Cycle

- A path visits every node exactly once
- Also NP-complete



Euler cycle

- A cycle visits every edge exactly once
- Famous “Seven Bridges of Königsberg”



Euler cycle

- A cycle visits every edge exactly once
- Famous “Seven Bridges of Königsberg”

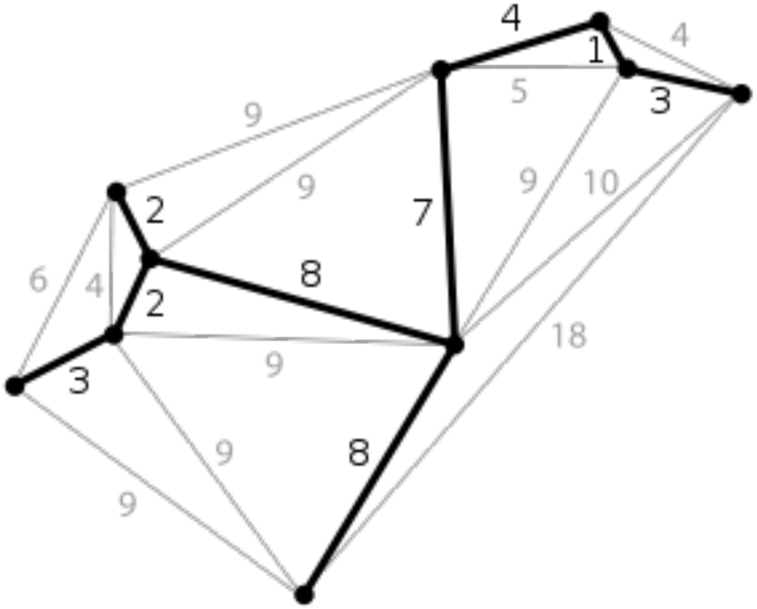
- Simple P-problem:
    - Euler cycle ⇔ the degree of every node is even
    - Also apply to multigraph (multiple edges between a pair of vertices)

Approximate solution

- k-approximate solution: a cycle that travels through every city once, and its length L:
    - $L \leq kL'$, where L’ is the real TSP solution

Minimum Spanning Tree

- A tree containing all vertices of V and having minimum total weights



Christofides algorithm

1. Find a minimum spanning tree T of G
2. Let O be the set of vertices with odd degrees in T, find a minimum perfect matching M in the subgraph induced by O
3. Combine T and M to form a multigraph H
4. Form an Euler Tour P in H
5. Make P Hamiltonian by skipping visited nodes.

Christofides algorithm

1. Find a minimum spanning tree T of G
2. Let O be the set of vertices with odd degrees in T, find a minimum perfect matching M in the subgraph induced by O
    - Since G is a complete graph, and |O| is even, there is always a perfect matching.
3. Combine T and M to form a multigraph H
4. Form an Euler Tour P in H
5. Make P Hamiltonian by skipping visited nodes.

Christofides algorithm

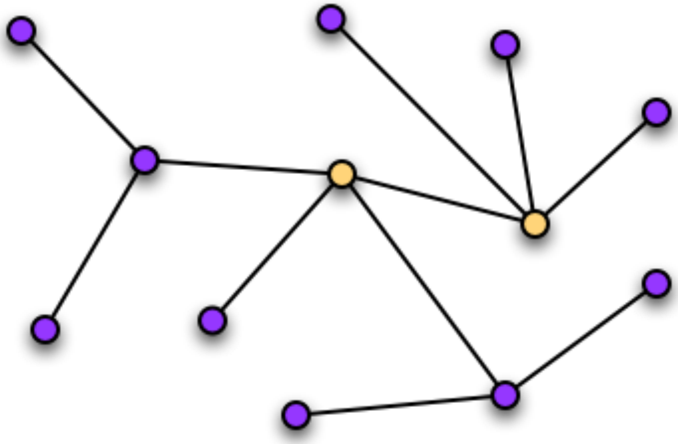
1. Find a minimum spanning tree T of G
2. Let O be the set of vertices with odd degrees in T, find a minimum perfect matching M in the subgraph induced by O
   - Since G is a complete graph, and |O| is even, there is always a perfect matching.
3. Combine T and M to form a multigraph H
   - The degree of every node in H is even, so there is a Euler cycle in H
4. Form an Euler cycle P in H
5. Make P Hamiltonian by skipping visited nodes.

Christofides algorithm

1. Find a minimum spanning tree T of G
2. Let O be the set of vertices with odd degrees in T, find a minimum perfect matching M in the subgraph induced by O
    - Since G is a complete graph, and |O| is even, there is always a perfect matching.
3. Combine T and M to form a multigraph H
    - The degree of every node in H is even, so there is a Euler cycle in H
4. Form an Euler cycle P in H
5. Make P Hamiltonian by skipping visited nodes.
    - Since G satisfies the triangle inequality, shortcutting vertices cannot increase the length

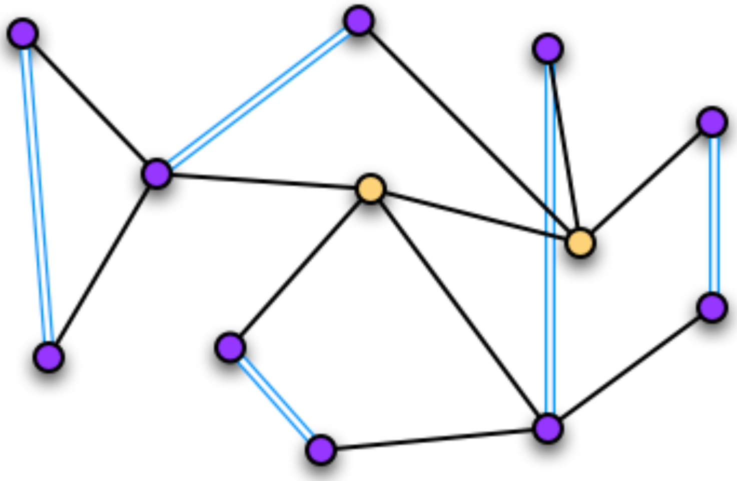
Example

- Vertices with odd degrees and vertices with even degrees



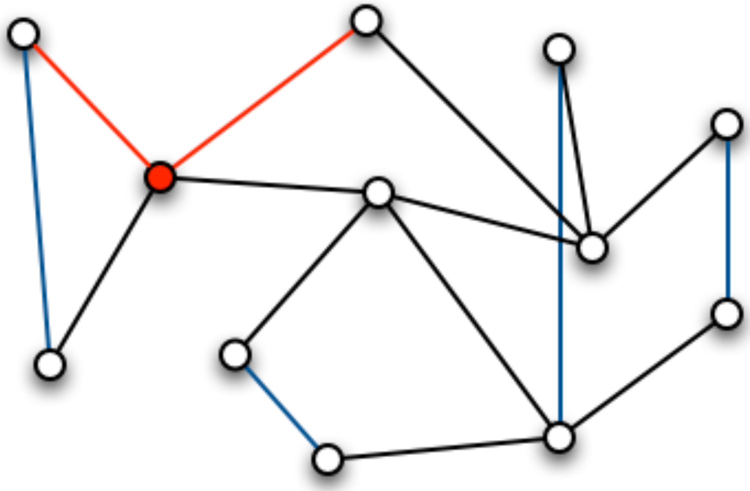
Example

- Find a perfect matching on odd vertices



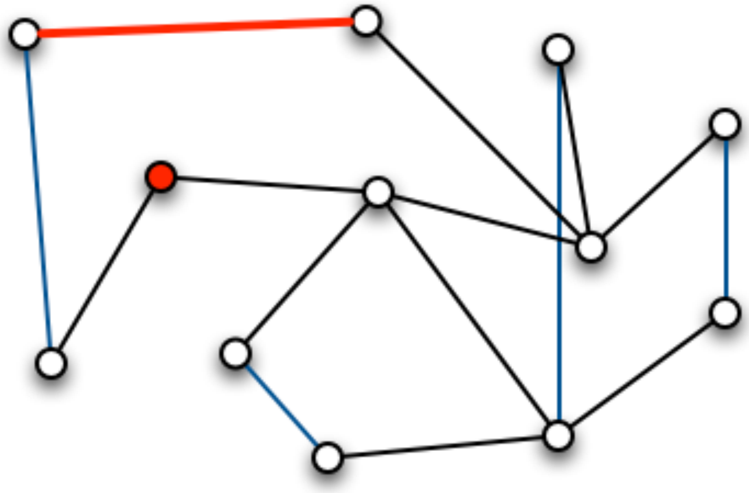
Example

- Then it must have an Euler cycle
- Shortcutting visited nodes



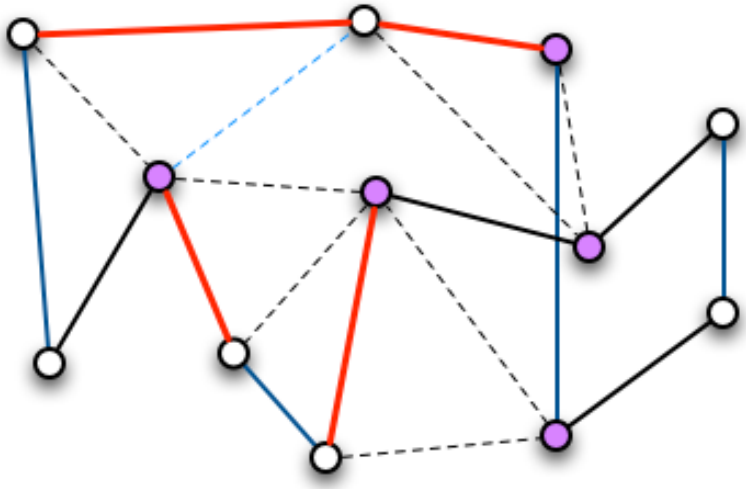
Example

- Then it must have an Euler cycle
- Shortcutting visited nodes



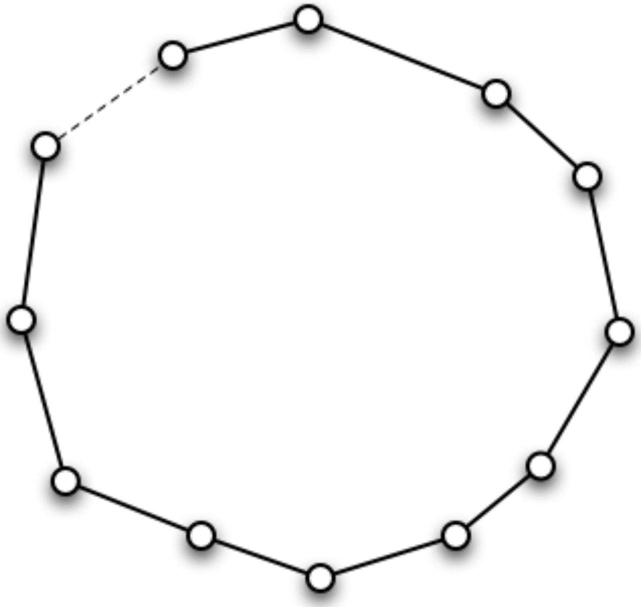
Example

- Then it must have an Euler cycle
- Shortcutting visited nodes
- Finally:



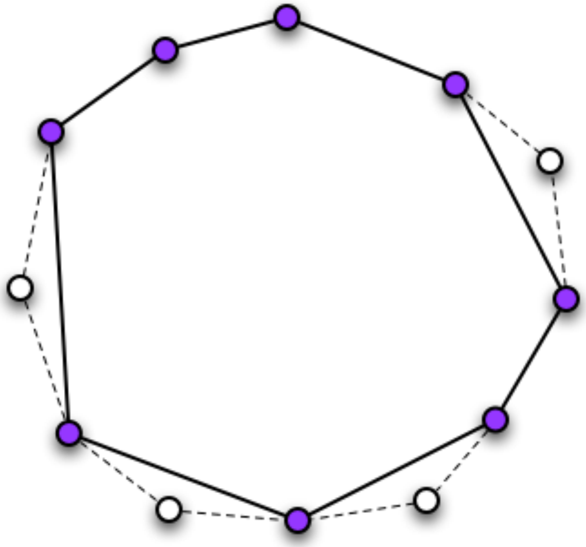
Approximate ratio: 1.5

- Weights of minimum spanning tree T is at most TSP
  - because TSP with one edge erased is a spanning tree



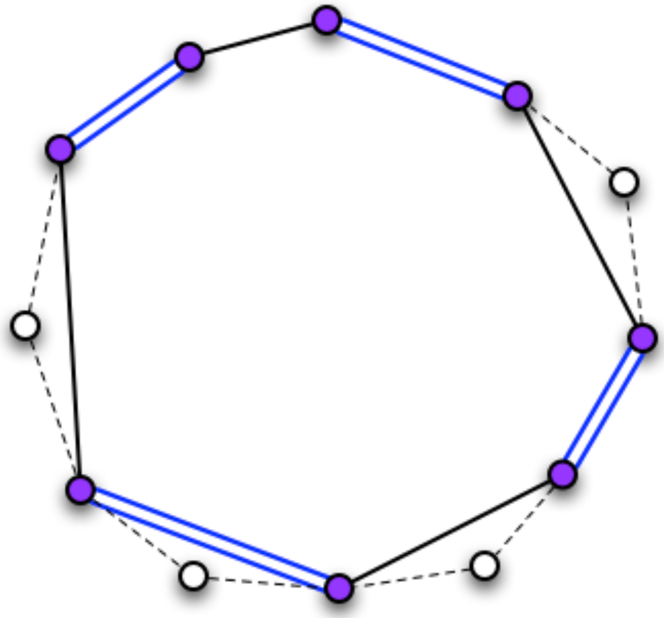
Approximate ratio: 1.5

- Weights of minimum spanning tree T is at most TSP
  - w(T)≤w(TSP)
- TSP on the subgraph induced by O is at most TSP on G
  - triangle inequlity



Approximate ratio: 1.5

- Weights of minimum spanning tree T is at most TSP
  - $w(T) \leq w(TSP)$
- TSP on the subgraph induced by O is at most TSP on G
  - $w(TSP^O) \leq w(TSP)$
  - $w(M) \leq \frac{1}{2} w(TSP^O)$



Approximate ratio: 1.5

- Weights of minimum spanning tree T is at most TSP
    - $w(T) \leq w(TSP)$
- TSP on the subgraph induced by O is at most TSP on G
    - $w(TSP^O) \leq w(TSP)$
    - $w(M) \leq \frac{1}{2} w(TSP^O)$
- Since shortcutting duplicated vertices will not increase length, the final path P
    - $w(P) \leq w(T) + w(M) \leq 1.5 w(TSP)$

Next lecture

- Dynamic connectivity